

z/OS Communications Server



IP Diagnosis Guide

Version 1 Release 4

z/OS Communications Server



IP Diagnosis Guide

Version 1 Release 4

Note:

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 749.

Third Edition (September 2002)

This edition applies to Version 1 Release 4 of z/OS (5694-A01) and Version 1 Release 4 of z/OS.e (5655-G52) and to all subsequent releases and modifications until otherwise indicated in new editions.

Publications are not stocked at the address given below. If you want more IBM® publications, ask your IBM representative or write to the IBM branch office serving your locality.

A form for your comments is provided at the back of this document. If the form has been removed, you may address comments to:

IBM Corporation
Software Reengineering
Department G71A/ Bldg 503
Research Triangle Park, NC 27709-9990
U.S.A.

If you prefer to send comments electronically, use one of the following methods:

Fax (USA and Canada):

1-800-254-0206

Internet e-mail:

usib2hpd@vnet.ibm.com

World Wide Web:

<http://www.ibm.com/servers/eserver/zseries/zos/webqs.html>

IBMLink™:

CIBMORCF at RALVM17

IBM Mail Exchange:

tkinlaw@us.ibm.com

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1994, 2002. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	xv
Tables	xvii
About this document	xix
Who should use this document	xix
Where to find more information	xix
Where to find related information on the Internet	xix
Accessing z/OS licensed documents on the Internet	xx
Using LookAt to look up message explanations	xxi
How to contact IBM service	xxi
z/OS Communications Server information	xxi
Summary of changes	xxix
<hr/> Part 1. General diagnosis information	1
Chapter 1. Overview of the diagnosis procedure	3
Chapter 2. Selecting tools and service aids	7
How do I know which tool or service aid to select?	7
What tools and service aids are available?	12
Dumps	12
Traces	13
First Failure Support Technology (FFST)	14
Display commands	16
System service aids	16
Submitting documentation electronically	17
Guidelines for machine-readable documentation	18
Necessary documentation	19
Chapter 3. Diagnosing abends, loops, and hangs	21
Analyzing abends	21
Analyzing loops	22
Analyzing hangs	23
Chapter 4. Diagnosing network connectivity problems	25
Communicating through the correct stack	25
Problems connecting to the server	26
Using Ping command	28
Using Netstat command	30
Using traceroute	40
Using SNMP remote PING	41
Diagnosing Sysplex Distributor problems	41
Documentation for the IBM Support Center	49
<hr/> Part 2. Traces and control blocks	51
Chapter 5. TCP/IP services traces and IPCS support	53
Component trace	53
Modifying options with the TRACE CT command	53
Displaying component trace status	55

Stopping a component trace	56
Obtaining component trace data with a dump.	56
Obtaining component trace data with an external writer	56
Tips for using component trace external writer	58
Formatting component traces	58
Event trace (SYSTCPIP) for TCP/IP stacks	62
Specifying trace options	62
Filter TCPIP CTRACE by IP address	70
Formatting event trace records for TCP/IP stacks	71
Socket API traces	74
Packet trace (SYSTCPDA) for TCP/IP stacks.	92
The trace process	92
Supported devices	93
Starting packet trace	93
Modifying options with VARY	94
Formatting packet traces using IPCS	95
Data trace (SYSTCPDA) for TCP/IP stacks	135
Starting data trace	135
Displaying data traces	135
Formatting data traces using IPCS	136
Intrusion Detection Services trace (SYSTCPIS)	136
Restrictions.	137
CTTRACE keywords on SYSTCPIS	137
SYSTCPIS OPTIONS	138
OPTIONS Keywords	142
IDS reports	152
OMPROUTE trace (SYSTCPRT)	166
RESOLVER trace (SYSTCPRE)	166
Configuration profile trace	166
 Chapter 6. IPCS subcommands for TCP/IP	 169
TCPIPICS	171
Command syntax	171
Parameters.	171
Symbols defined	173
TCPIPICS subcommands.	173
TCPIPICS API	173
TCPIPICS CONFIG	175
TCPIPICS CONNECTION	177
TCPIPICS COUNTERS	178
TCPIPICS DUAF	192
TCPIPICS DUCB	195
TCPIPICS FIREWALL	200
TCPIPICS FRCA	203
TCPIPICS HASH	205
TCPIPICS HEADER.	209
TCPIPICS HELP	210
TCPIPICS LOCK	211
TCPIPICS MAP	213
TCPIPICS MTABLE	214
TCPIPICS POLICY	215
TCPIPICS PROFILE	217
TCPIPICS PROTOCOL	219
TCPIPICS RAW	222
TCPIPICS ROUTE	224
TCPIPICS SOCKET.	225

TCIPCS STATE	227
TCIPCS STORAGE	238
TCIPCS STREAM.	239
TCIPCS TCB	241
TCIPCS TELNET	243
TCIPCS TIMER	244
TCIPCS TRACE	246
TCIPCS TREE	250
TCIPCS TSDB	253
TCIPCS TSDX	254
TCIPCS TSEB	255
TCIPCS UDP	256
TCIPCS VMCF	258
TCIPCS XCF	260
ERRNO	262
Syntax	262
Parameters.	262
Sample output	263
ICMPHDR	264
Syntax	264
Parameters.	264
Sample output	265
IPHDR	265
Syntax	265
Parameters.	266
Sample output	266
RESOLVER	267
Syntax	267
Parameters.	268
Sample output	269
SETPRINT	272
Syntax	272
Parameters.	272
Sample output	272
SKMSG	273
Syntax	273
Parameters.	273
Sample Output	274
TCPHDR	274
Syntax	274
Parameters.	275
Sample output	275
TOD	276
Syntax	276
Parameters.	276
Sample output	276
UDPHDR	277
Syntax	277
Parameters.	277
Sample output	278
Installing TCP/IP IPCS subcommands	278
Entering TCP/IP IPCS subcommands	279
Using the batch option	281

Chapter 7. Diagnosing line print requester and daemon (LPR and LPD) problems	285
Diagnosing LPR client and LPD server problems	285
Abends	285
Timeouts, hangs, and waits	286
Incorrect output	286
LPR client traces	288
Activating LPR client traces	288
Client trace output	288
LPD server traces	294
Activating server traces	294
Server trace output	295
Chapter 8. Diagnosing File Transfer Protocol (FTP) problems	309
FTP server	309
Structural overview	309
Definitions and setup	309
Error exit codes	310
Name considerations for z/OS UNIX FTP	310
Translation and data conversion support	311
Common z/OS UNIX FTP problems	313
Diagnosing FTP server problems with traces	326
Logging FTP server activity	334
Documenting server problems	335
FTP client	335
Execution environments	335
Setup	336
Naming considerations	336
Translation and data conversion support	336
File tagging support	338
Common problems	341
DB2 query support	345
Diagnosing FTP client problems with tracing	347
Documenting FTP client problems	348
Chapter 9. Diagnosing z/OS UNIX Telnet daemon problems	349
Common problems	349
Debug traces	349
Debug trace flows (netdata and ptydata)	350
Debug trace examples (-t -D all)	350
Cleaning up the utmp entries left from dead processes	364
Chapter 10. Diagnosing Telnet problems	365
General Telnet server information	365
Telnet server definitions	365
Diagnosing Telnet server problems	365
Abends (server)	366
Logon problems (server)	366
Session hangs (server)	368
Incorrect output (server)	369
Session outages (server)	370
Special considerations when using SSL encryption support	372
Telnet Component Trace data	373
General Telnet client information	373
Telnet client definitions	373
Diagnosing Telnet client problems	373

Abends (client)	374
Session hangs (client)	374
Incorrect output (client)	375
Telnet client traces	376
Starting Telnet client traces	376
Trace example (client)	377
Telnet commands and options	380
 Chapter 11. Diagnosing Simple Mail Transfer Protocol (SMTP) problems	383
Sender SMTP	383
Receiver SMTP	383
SMTP environment	383
SMTP definitions	384
Diagnosing SMTP problems	384
Abends	384
Spooling problems	384
SMTP does not deliver mail	385
SMTP loop	386
Mail item has incorrect output	386
Forcing resolution of queued mail	387
ADDRBLOK data set	388
SMTP RESOLVER trace	390
 Chapter 12. Diagnosing z/OS UNIX sendmail and popper problems	393
Diagnostic aids for sendmail	393
Debugging switches	393
Additional diagnostic aids	398
Diagnostic aids for popper	400
 Chapter 13. Diagnosing SNALINK LU0 problems	403
Definitions	403
Problem diagnosis	403
Abends	404
Session hangs	404
Session outages	405
Traces	406
Using IP packet trace	406
SNALINK LU0 DEBUG trace	413
 Chapter 14. Diagnosing SNALINK LU6.2 problems	417
Setting up a SNALINK LU6.2 network	417
Common configuration mistakes	419
Diagnosing problems	419
Quick checklist for common problems	419
Problems starting the SNALINK LU6.2 address space	420
DLC connection problems	422
Network connection establishment problems	424
Network connection loss problems	426
Data loss problems	427
Data corruption problems	428
Documentation references for problem diagnosis	429
Using NETSTAT	429
Using the SNALINK LU6.2 subcommand	430
Useful VTAM operations	431
Traces	433
Using SNALINK LU6.2 internal traces	433

Using IP packet trace	437
TCP/IP internal traces	438
VTAM buffer traces	439
Finding abend and sense code documentation	439
Finding error message documentation	439

Chapter 15. Diagnosing name server and dynamic domain name server

(DDNS) problems	441
Diagnosing name server problems	441
Determining the name server version	441
Checking messages sent to the operator's console	443
Checking the log messages	443
Tools for Querying the name server	445
Using the debug option with the name server	445
Debugging with a resolver directive	447
Using the remote name daemon control (rndc) program, BIND 9 name server only	447
Using name server signals	448
Interpreting BIND 4.9.3 name server statistics	449
Statistics file for the Bind 9 name server	450
Using the nsupdate command	451
Return codes	451
Using component trace	452
Diagnosing problems with connection optimization	452
Addresses not being returned	452
Connection problems	453

Chapter 16. Diagnosing REXEC, REXECD, and RSH problems

General information about REXEC and RSH	455
Documentation for REXEC problem diagnosis	455
TSO console log	455
Activating the REXEC debug trace	456
RExec trace example and explanation	456
RSH trace example and explanation	458
General information about REXECD	459
Documentation for REXECD problem diagnosis	459
MVS system console log	459
Starting REXECD server traces	459
Example of an REXECD trace of a client using the SEND command	460
Example trace of an RSH client using the SEND command	460

Chapter 17. Diagnosing z/OS UNIX REXEC, REXECD, and RSHD problems

Setting up the inetd configuration file	463
Diagnosing z/OS UNIX REXEC	464
Activating the z/OS UNIX REXEC debug trace	464
z/OS UNIX REXEC trace example and explanation	464
Diagnosing z/OS UNIX REXECD	465
Activating the z/OS UNIX REXECD debug trace	465
z/OS UNIX REXECD trace example and explanation	465
Diagnosing z/OS UNIX RSHD	466
Activating the z/OS UNIX RSHD debug trace	466
z/OS UNIX RSHD trace example and explanation	466

Chapter 18. Diagnosing network database system (NDB) problems

Documentation for NDB problem diagnosis	468
Definitions	468

Diagnosing NDB problems	469
NDB trace examples and explanations.	470
Chapter 19. Diagnosing X Window System and OSF/Motif problems . . .	483
Trace output when XWTRACE=2.	483
Trace output when XWTRACELC=2	484
Chapter 20. Diagnosing Simple Network Management Protocol (SNMP)	
problems	487
Overview	487
Management information base (MIB)	487
PDUs	487
Functional components	488
Definitions	489
osnmp	489
SNMP agent	489
TCP/IP subagent.	490
OMPROUTE subagent	490
SLA subagent	490
SNMP socket call settings	490
Trap forwarder daemon	491
Diagnosing SNMP problems	491
Abends	491
SNMP connection problems.	492
Incorrect output	498
No response from the SNMP agent	503
Report received from SNMP agent	504
I/O error for SNMP PING.	505
Traps not forwarded by trap forwarder daemon	505
Incorrect address in forwarded trap	506
SNMP traces	507
Starting manager traces	507
Starting SNMP agent traces	508
Starting TCP/IP subagent traces	509
Starting OMPROUTE subagent traces	509
Starting SLA subagent traces	509
Starting TRAPFWD traces	509
Trace examples and explanations	510
Chapter 21. Diagnosing Policy Agent problems	531
Overview	531
QoS policy scopes	531
Gathering diagnostic information	532
Diagnosing Policy Agent problems	534
Initialization problems	534
Policy definition problems	535
LDAP object retrieval problems	537
LDAP object storage problems	538
Policy Agent/Sysplex distribution problems	540
Example log file	541
Chapter 22. Diagnosing RSVP agent problems.	561
Overview	561
Reservation types, styles, and objects	561
Policies and RSVP processing.	563
Gathering diagnostic information	564

Diagnosing RSVP agent problems	564
Initialization problems	564
Application problems	565
Policy problems	565
Example log file	565
Chapter 23. Diagnosing intrusion detection problems	575
Overview	575
Diagnosing IDS policy problems	575
Diagnosing IDS output problems	575
IDS SyslogD output is missing.	575
IDS console output	576
IDS packet trace output	576
Unusual conditions	576
Diagnosing TRMD problems	578
Documentation for the IBM Software Support Center	579
Chapter 24. Diagnosing OROUTED problems	581
Overview	581
Definitions	582
Diagnosing OROUTED problems.	582
Abends	583
OROUTED connection problems	583
Analyzing routing failures.	584
Incorrect output	586
Session outages	586
OROUTED traces and debug information.	587
Starting OROUTED traces from the z/OS UNIX shell	588
Starting OROUTED traces from an MVS catalogued procedure	589
Where to send OROUTED trace output	590
Stopping OROUTED	590
Changing trace and debug levels with MODIFY	591
OROUTED trace example and explanation	591
Documentation for the IBM Software Support Center	599
Chapter 25. Diagnosing OMPROUTE problems	601
Overview	601
Definitions	602
Diagnosing OMPROUTE problems	603
Abends	603
OMPROUTE connection problems	603
Routing failures	603
OMPROUTE traces and debug information	605
Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell	606
Starting OMPROUTE tracing and debugging from an MVS cataloged procedure or AUTOLOG	606
Starting OMPROUTE tracing and debugging using the MODIFY command	607
Destination of OMPROUTE trace and debug output	607
Sample OMPROUTE trace output	608
TCP/IP services component trace for OMPROUTE	615
Specifying trace options	616
Formatting OMPROUTE trace records.	618
Chapter 26. Diagnosing NCPROUTE problems.	619
Definitions	621

Diagnosing NCPROUTE problems	622
Abends	622
Connection problems	622
Analyzing routing failures.	625
Incorrect output	627
Session outages	628
NCPROUTE traces	629
Activating NCPROUTE global traces	629
Activating NCPROUTE selective traces	630
NCPROUTE trace example and explanation	631
 Chapter 27. Diagnosing X.25 NPSI problems	643
Operation	644
Configuration requirements	645
RACF/Security Manager requirement	645
VTAM considerations	645
NPSI considerations	645
Sources of diagnostic information	646
X.25 trace examples	646
Normal incoming call, TRACE OFF	647
Normal incoming call, TRACE DATA	647
Normal outgoing call, TRACE CONTROL.	648
Results of LIST command	648
Termination by TCPIP STOP device.	649
Logon problems	649
Session hangs	650
Helpful hints	650
Documentation requirements	651
 Chapter 28. Diagnosing IMS problems	653
Setting up the IMS TCP/IP services socket interface system.	654
Common configuration mistakes	656
Quick checklist for common problems	656
Component problems	657
Connection problems	658
Error message and return code problems.	661
Socket data protocol problems.	661
IMS transaction build problems	663
IMS database problems	664
Documentation references for problem diagnosis	666
Traces	666
Using NETSTAT	667
Where to find return code documentation.	668
Where to find error message documentation	669
 Chapter 29. Diagnosing restartable VMCF/TNF problems.	671
VMCF or TNF fail to initialize	671
Abends 0D5 and 0D6	671
No response to commands	671
VMCF or TNF will not stop	671
 Chapter 30. Diagnosing problems with CICS	673
Diagnostic data	673
Initialization problems	673
CICS sockets interface not initialized	674
CICS listener not initialized	674

No CICS sockets messages issued	674
TCP/IP clients unable to connect	674
Child-server transactions not starting	675
CICS sockets application problems	675
Hung CICS tasks	675
Hung CICS region	675
Errors on socket calls	675
CICS shutdown hangs.	675
CICS sockets control blocks	676
Task interface element	676
Global work area.	676
CICS trace	676
Chapter 31. Diagnosing problems with Express Logon	679
Analyzing start problems with the DCAS	680
Analyzing client interface problems	680
Chapter 32. Diagnosing resolver problems	683
Steps for resolving the hostname.	683
TRACE RESOLVER	684
CTRACE — RESOLVER.	695
Chapter 33. Diagnosing Simple Network Time Protocol (SNTP) problems	699
Activating the SNTPD debug trace	699
Abends	699
Steps for stopping SNTPD	699
Sample SNTPD debug output	699

Part 4. Appendixes 701

Appendix A. First Failure Support Technology (FFST)	703
FFST probe index	703
FFST probe information	703
FFST probe naming conventions	704
FFST probe descriptions	704
Appendix B. Overview of internetworking	713
Maximum transmission unit (MTU)	714
Fiber Distributed Data Interface (FDDI)	715
Token-Ring IEEE 802.5	716
IEEE 802.3	717
Ethernet — DIX V2	717
Subnetwork Access Protocol (SNAP)	718
IP routing	719
Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6)	719
Internet Protocol Version 4 (IPv4)	719
Internet Protocol Version 6 (IPv6)	720
Direct routing	722
Indirect routing	723
Simplified IP datagram routing algorithm	723
IPv4 Subnetting	724
IPv6 Prefixes	725
Simplified IP datagram routing algorithm with subnets	726
Static routing	727
Dynamic routing	727
IPv4	727

I	IPv6	728
	Appendix C. How to read a syntax diagram	729
	Symbols and punctuation	729
	Parameters.	729
	Syntax examples.	729
	Longer than one line	730
	Required operands	730
	Choose one required item from a stack	730
	Optional values	730
	Choose one optional operand from a stack	730
	Repeating an operand.	730
	Selecting more than one operand	731
	Nonalphanumeric characters	731
	Blank spaces in syntax diagrams.	731
	Default operands.	731
	Variables.	731
	Syntax fragments	731
	Appendix D. Related protocol specifications (RFCs)	733
I	Draft RFCs	740
	Appendix E. Information APARs	743
	Information APARs for IP documents	743
	Information APARs for SNA documents	744
	Other information APARs.	744
	Appendix F. Accessibility	747
	Using assistive technologies	747
	Keyboard navigation of the user interface.	747
	Notices	749
	Trademarks.	752
	Index	755
I	Communicating Your Comments to IBM	765

Figures

1. Overview of the diagnosis procedure	4
2. Example of output from the IPCS SYSTRACE command	23
3. Procedure for diagnosing server connection problems	26
4. IPCS CTRACE	59
5. SYS1.PARMLIB member CTIEZB00.	63
6. Start of component trace full format	73
7. Component trace full format showing character interpretation of fields	74
8. TCP/IP networking API relationship on z/OS.	75
9. Data trace record.	91
10. SOCKAPI trace record.	92
11. Control and data flow in the IP packet tracing facility.	93
12. Data trace: Single entry	135
13. Data trace: Multiple entries.	136
14. SYS1.PARMLIB member CTIIDS00	137
15. IPCS primary option menu.	280
16. IPCS subcommand entry panel with a TCP/IP IPCS subcommand entered	280
17. Main menu for TCP/IP IPCS subcommands.	281
18. Example of LPR trace output	289
19. Example of LPR trace with filter x option	291
20. Example of LPR output with unknown printer	292
21. Example of LPR trace with JNUM, LANDSCAPE, and TRACE options	293
22. Example of LPR trace with XLATE option	294
23. Example of LPD trace specified with the DEBUG option	295
24. Example of an LPD server trace of a failing job	301
25. Example of an LPD server trace for a remote print request	304
26. Trace between the Telnet client, parent, and child	350
27. z/OS UNIX Telnet trace using -t -D all.	351
28. Telnet client trace	377
29. SMTP environment	383
30. Example of RESOLVER trace output	391
31. Invoking TRCFMT from TSO	411
32. IP packet trace output for SNALINK LU0	412
33. Invoking TRCFMT in a batch job	413
34. Example of a SNALINK LU0 DEBUG trace.	414
35. Components of a SNALINK LU6.2 connection on MVS	417
36. Sample MVS System Console Messages on SNALINK LU6.2 Address Space Startup	418
37. NETSTAT DEVLINKS output example.	429
38. LIST MODIFY subcommand output example	430
39. DISPLAY subcommand output example for connectable LU	432
40. DISPLAY subcommand output example for active LU	433
41. SNALINK LU6.2 internal trace output	435
42. A CTRACE formatted packet trace record	438
43. Remote execution protocol principle	455
44. Example of an REXEC trace	457
45. Example of an RSH trace	458
46. Example of an REXECD trace of a client using a SEND command	460
47. Example of a trace of an RSH client using a SEND command.	461
48. Adding applications to /etc/inetd.conf	463
49. Setting traces in /etc/inetd.conf	464
50. Components of the network database system.	467
51. NDB port manager trace with two NDB servers started and one client invoked.	471
52. NDB port client trace with two NDB servers started and one client invoked	473
53. Example of X Application trace output when XWTRACE=2	484

54.	Example of X Application trace output when XWTRACELC=2	485
55.	SNMP agent response trace	511
56.	SNMP agent trace of unsuccessful initialization	511
57.	SNMP messages and agent trace for nonmatching key	511
58.	SNMP messages and agent trace when data not in defined view	512
59.	SNMP subagent trace	512
60.	SNMP query engine traces.	514
61.	SNMP IUCV communication traces	526
62.	TRAPFWD trace	530
63.	Policy Agent	543
64.	RSVP Agent processing log	566
65.	OROUTED environment.	582
66.	Sample OROUTED environment	588
67.	Example of an OROUTED trace.	592
68.	Sample OMPROUTE Trace Output.	609
69.	SYS1.PARMLIB member CTIORA00	617
70.	NCPROUTE environment	619
71.	NCPROUTE trace	632
72.	X.25 NPSI environment	644
73.	Components of the IMS TCP/IP services socket interface system	653
74.	Routers and bridges within an internet	714
75.	Relationship of MTU to frame size	715
76.	Format of an IEEE 802.5 token-ring frame	716
77.	Format of an IEEE 802.3 frame	717
78.	Format of an Ethernet V2 frame.	718
79.	SNAP header	718
I 80.	Classes of IPv4 addresses.	719
I 81.	Determining the class of an IPv4 address	720
82.	Routing and bridging	722
83.	General IP routing algorithm	723
84.	Subnetting scheme	724
85.	Routing algorithm with subnets	726
86.	Example of resolving a subnet route	727

Tables

1.	Selecting a dump	7
2.	Selecting a trace	8
3.	Selecting a service aid	11
4.	Description of dumps	12
5.	Description of traces	13
6.	Description of service aids	16
7.	TCP/IP component name and release level	20
8.	Types of abends	21
I 9.	Diagnostic commands	25
10.	Diagnosis of a timeout	29
11.	Trace options	66
12.	IP address and port filtering effect on different types of socket API calls	79
13.	TCP/IP IPCS commands - showing the TCPIPCS commands first, followed by the general commands.	169
14.	Target data sets for TCP/IP IPCS subcommands	278
15.	SQL problems generating 55x replies (FTP Server).	324
16.	Other SQL problems (FTP Server)	325
17.	SQL problems generating 55x replies (FTP Client)	346
18.	Other SQL problems (FTP Client)	347
19.	Debug trace options	350
20.	Telnet login problems	366
21.	Incorrect output types for Telnet	369
22.	Telnet commands from RFC 854	381
23.	Telnet command options from RFC 1060	381
24.	Format of Record 1 of an SMTP ADDRBLK data set	388
25.	Format of Record 2 (for an unresolved from record) of an SMTP ADDRBLK data set	389
26.	Format of Record 2 (for a resolved from record) of an SMTP ADDRBLK data set	389
27.	Format of Record 3 (for an unresolved from record) of an SMTP ADDRBLK data set	389
28.	Debugging switches by category	393
I 29.	qf File code letters	399
30.	Format of a SNALINK trace table entry	415
I 31.	BIND 9 debug information	446
32.	Configuration files and security types	490
33.	OROUTED incorrect output	586
34.	OROUTED session outages	587
35.	OMPROUTE trace options	618
36.	NCPROUTE connection problems	623
37.	Diagnostic steps for NCPROUTE connection problems	623
38.	NCPROUTE routing failures	626
39.	Diagnostic steps for NCPROUTE routing failures	626
40.	NCPROUTE incorrect output	627
41.	Diagnostic steps for NCPROUTE incorrect output	628
42.	NCPROUTE session outages.	629
43.	Diagnostic steps for NCPROUTE session outages	629
44.	FFST probes	703
45.	FFST naming conventions	704
46.	IOCTL enablement probes	704
47.	Infrastructure services probes.	704
48.	FFST probes for Pascal API	705
49.	PFS IOCTL probes	709
50.	Telnet transform probes	710
51.	FFST probes for Telnet SRV	710
52.	Configuration services probes	710

53.	TCP/IP Base probes	710
54.	Transmission Control Protocol probes.	711
55.	Update Datagram Protocol Layer probes.	711
56.	Streams probes	711
57.	Raw IP Layer probes	712
58.	FFST probes for Internet Protocol	712
59.	XCF probes	712
60.	Relationship between RC field and maximum l-field value	717
61.	IP information APARs.	743
62.	SNA information APARs	744
63.	Non-document information APARs	745

About this document

This document tells you how to diagnose and report problems occurring in the IBM z/OS™ Transmission Control Protocol/Internet Protocol (TCP/IP). Additional information is provided for diagnosing problems with selected applications that are part of z/OS Communications Server V1R4 (z/OS CS). The information in this document supports both IPv6 and IPv4. Unless explicitly noted, information describes IPv4 networking protocol. IPv6 support is qualified within the text.

Use this document to perform the following tasks:

- Diagnose and solve problems in an z/OS CS installation
- Describe problems to the IBM Software Support Center and document the problems appropriately

This document supports z/OS.e™.

Who should use this document

Use this document if you are a system programmer to diagnose problems with TCP/IP or to diagnose problems with z/OS CS components.

To use this document, you should be familiar with z/OS TCP/IP Services and the TCP/IP suite of protocols.

Where to find more information

This section contains:

- Pointers to information available on the Internet
- Information about licensed documentation
- Information about LookAt, the online message tool
- A set of tables that describes the documents in the z/OS Communications Server (z/OS CS) library, along with related publications

Where to find related information on the Internet

z/OS

- <http://www.ibm.com/servers/eserver/zseries/zos/>

z/OS Internet Library

- <http://www.ibm.com/servers/eserver/zseries/zos/bkserv/>

IBM Communications Server product

- <http://www.software.ibm.com/network/commserver/>

IBM Communications Server product support

- <http://www.software.ibm.com/network/commserver/support/>

IBM Systems Center publications

- <http://www.redbooks.ibm.com/>

IBM Systems Center flashes

- <http://www-1.ibm.com/support/techdocs/atsmastr.nsf>

RFCs

- <http://www.ietf.org/rfc.html>

RFC drafts

- <http://www.ietf.org/ID.html>

Information about Web addresses can also be found in information APAR II11334.

DNS web sites

For more information about DNS, see the following USENET news groups and mailing:

USENET news groups:

`comp.protocols.dns.bind`

For BIND mailing lists, see:

- <http://www.isc.org/ml-archives/>
 - BIND Users
 - Subscribe by sending mail to `bind-users-request@isc.org`.
 - Submit questions or answers to this forum by sending mail to `bind-users@isc.org`.
 - BIND 9 Users (Note: This list may not be maintained indefinitely.)
 - Subscribe by sending mail to `bind9-users-request@isc.org`.
 - Submit questions or answers to this forum by sending mail to `bind9-users@isc.org`.

For definitions of the terms and abbreviations used in this document, you can view or download the latest *IBM Glossary of Computing Terms* at the following Web address:

<http://www.ibm.com/ibm/terminology>

Note: Any pointers in this publication to Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

<http://www.ibm.com/servers/resourceLink>

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code.

To obtain your IBM Resource Link user ID and password, log on to:

<http://www.ibm.com/servers/resourceLink>

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for most messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

<http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/>

or from anywhere in z/OS where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS). You can also download code from the *z/OS Collection* (SK3T-4269) and the LookAt Web site that will allow you to access LookAt from a handheld computer (Palm Pilot VIIx suggested).

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the **News** section on the LookAt Web site.

Some messages have information in more than one document. For those messages, LookAt displays a list of documents in which the message appears.

How to contact IBM service

For immediate assistance, visit this Web site:

<http://www.software.ibm.com/network/commserver/support/>

Most problems can be resolved at this Web site, where you can submit questions and problem reports electronically, as well as access a variety of diagnosis information.

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-237-5511). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m. – 5:00 p.m., local customer time).

Outside of the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

If you would like to provide feedback on this publication, see “Communicating Your Comments to IBM” on page 765.

z/OS Communications Server information

This section contains descriptions of the documents in the z/OS Communications Server library.

z/OS Communications Server publications are available:

- Online at the z/OS Internet Library web page at <http://www.ibm.com/servers/eserver/zseries/zos/bkserv>
- In softcopy on CD-ROM collections.

Softcopy information

Softcopy publications are available in the following collections:

Titles	Order Number	Description
<i>z/OS V1R4 Collection</i>	SK3T-4269	This is the CD collection shipped with the z/OS product. It includes the libraries for z/OS V1R4, in both BookManager® and PDF formats.
<i>z/OS Software Products Collection</i>	SK3T-4270	This CD includes, in both BookManager and PDF formats, the libraries of z/OS software products that run on z/OS but are not elements and features, as well as the <i>Getting Started with Parallel Sysplex®</i> bookshelf.
<i>z/OS V1R4 and Software Products DVD Collection</i>	SK3T-4271	This collection includes the libraries of z/OS (the element and feature libraries) and the libraries for z/OS software products in both BookManager and PDF format. This collection combines SK3T-4269 and SK3T-4270.
<i>z/OS Licensed Product Library</i>	SK3T-4307	This CD includes the licensed documents in both BookManager and PDF format.
<i>System Center Publication IBM S/390® Redbooks™ Collection</i>	SK2T-2177	This collection contains over 300 ITSO redbooks that apply to the S/390 platform and to host networking arranged into subject bookshelves.

z/OS Communications Server library

z/OS V1R4 Communications Server documents are available on the CD-ROM accompanying z/OS (SK3T-4269 or SK3T-4307). Unlicensed documents can be viewed at the z/OS Internet library site.

Updates to documents are available on RETAIN® and in information APARs (info APARs). See Appendix E, "Information APARs" on page 743 for a list of the documents and the info APARs associated with them.

- Info APARs for OS/390® documents are in the document called *OS/390 DOC APAR and PTF ++HOLD Documentation* which can be found at http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IDDOCMST/CCONTENTS.
- Info APARs for z/OS documents are in the document called *z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation* which can be found at http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS.

Planning and migration:

Title	Number	Description
<i>z/OS Communications Server: SNA Migration</i>	GC31-8774	This document is intended to help you plan for SNA, whether you are migrating from a previous version or installing SNA for the first time. This document also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with SNA.
<i>z/OS Communications Server: IP Migration</i>	GC31-8773	This document is intended to help you plan for TCP/IP Services, whether you are migrating from a previous version or installing IP for the first time. This document also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with TCP/IP Services.

Title	Number	Description
<i>z/OS Communications Server: IPv6 Network and Application Design Guide</i>	SC31-8885	This document is a high-level introduction to IPv6. It describes concepts of z/OS Communications Server's support of IPv6, coexistence with IPv4, and migration issues.

Resource definition, configuration, and tuning:

Title	Number	Description
<i>z/OS Communications Server: IP Configuration Guide</i>	SC31-8775	This document describes the major concepts involved in understanding and configuring an IP network. Familiarity with the z/OS operating system, IP protocols, z/OS UNIX® System Services, and IBM Time Sharing Option (TSO) is recommended. Use this document in conjunction with the <i>z/OS Communications Server: IP Configuration Reference</i> .
<i>z/OS Communications Server: IP Configuration Reference</i>	SC31-8776	This document presents information for people who want to administer and maintain IP. Use this document in conjunction with the <i>z/OS Communications Server: IP Configuration Guide</i> . The information in this document includes: <ul style="list-style-type: none"> • TCP/IP configuration data sets • Configuration statements • Translation tables • SMF records • Protocol number and port assignments
<i>z/OS Communications Server: SNA Network Implementation Guide</i>	SC31-8777	This document presents the major concepts involved in implementing an SNA network. Use this document in conjunction with the <i>z/OS Communications Server: SNA Resource Definition Reference</i> .
<i>z/OS Communications Server: SNA Resource Definition Reference</i>	SC31-8778	This document describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA. Use this document in conjunction with the <i>z/OS Communications Server: SNA Network Implementation Guide</i> .
<i>z/OS Communications Server: SNA Resource Definition Samples</i>	SC31-8836	This document contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.
<i>z/OS Communications Server: AnyNet SNA over TCP/IP</i>	SC31-8832	This guide provides information to help you install, configure, use, and diagnose SNA over TCP/IP.
<i>z/OS Communications Server: AnyNet Sockets over SNA</i>	SC31-8831	This guide provides information to help you install, configure, use, and diagnose sockets over SNA. It also provides information to help you prepare application programs to use sockets over SNA.
<i>z/OS Communications Server: IP Network Print Facility</i>	SC31-8833	This document is for system programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP Services.

Operation:

Title	Number	Description
<i>z/OS Communications Server: IP User's Guide and Commands</i>	SC31-8780	This document describes how to use TCP/IP applications. It contains requests that allow a user to log on to a remote host using Telnet, transfer data sets using FTP, send and receive electronic mail, print on remote printers, and authenticate network users.
<i>z/OS Communications Server: IP System Administrator's Commands</i>	SC31-8781	This document describes the functions and commands helpful in configuring or monitoring your system. It contains system administrator's commands, such as TSO NETSTAT, PING, TRACERTE and their UNIX counterparts. It also includes TSO and MVS™ commands commonly used during the IP configuration process.
<i>z/OS Communications Server: SNA Operation</i>	SC31-8779	This document serves as a reference for programmers and operators requiring detailed information about specific operator commands.
<i>z/OS Communications Server: Quick Reference</i>	SX75-0124	This document contains essential information about SNA and IP commands.

Customization:

Title	Number	Description
<i>z/OS Communications Server: SNA Customization</i>	LY43-0092	<p>This document enables you to customize SNA, and includes the following:</p> <ul style="list-style-type: none"> • Communication network management (CNM) routing table • Logon-interpret routine requirements • Logon manager installation-wide exit routine for the CLU search exit • TSO/SNA installation-wide exit routines • SNA installation-wide exit routines

Writing application programs:

Title	Number	Description
<i>z/OS Communications Server: IP Application Programming Interface Guide</i>	SC31-8788	This document describes the syntax and semantics of program source code necessary to write your own application programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this document to adapt your existing applications to communicate with each other using sockets over TCP/IP.
<i>z/OS Communications Server: IP CICS Sockets Guide</i>	SC31-8807	This document is for programmers who want to set up, write application programs for, and diagnose problems with the socket interface for CICS® using z/OS TCP/IP.
<i>z/OS Communications Server: IP IMS Sockets Guide</i>	SC31-8830	This document is for programmers who want application programs that use the IMS™ TCP/IP application development services provided by IBM's TCP/IP Services.

Title	Number	Description
<i>z/OS Communications Server: IP Programmer's Reference</i>	SC31-8787	This document describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the z/OS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.
<i>z/OS Communications Server: SNA Programming</i>	SC31-8829	This document describes how to use SNA macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain.
<i>z/OS Communications Server: SNA Programmer's LU 6.2 Guide</i>	SC31-8811	This document describes how to use the SNA LU 6.2 application programming interface for host application programs. This document applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this document.)
<i>z/OS Communications Server: SNA Programmer's LU 6.2 Reference</i>	SC31-8810	This document provides reference material for the SNA LU 6.2 programming interface for host application programs.
<i>z/OS Communications Server: CSM Guide</i>	SC31-8808	This document describes how applications use the communications storage manager.
<i>z/OS Communications Server: CMIP Services and Topology Agent Guide</i>	SC31-8828	This document describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The document provides guide and reference information about CMIP services and the SNA topology agent.

Diagnosis:

Title	Number	Description
<i>z/OS Communications Server: IP Diagnosis</i>	GC31-8782	This document explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.
<i>z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures and z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT</i>	LY43-0088 LY43-0089	These documents help you identify an SNA problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation.
<i>z/OS Communications Server: SNA Data Areas Volume 1 and z/OS Communications Server: SNA Data Areas Volume 2</i>	LY43-0090 LY43-0091	These documents describe SNA data areas and can be used to read an SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

Messages and codes:

Title	Number	Description
<i>z/OS Communications Server: SNA Messages</i>	SC31-8790	This document describes the ELM, IKT, IST, ISU, IUT, IVT, and USS messages. Other information in this document includes: <ul style="list-style-type: none"> • Command and RU types in SNA messages • Node and ID types in SNA messages • Supplemental message-related information
<i>z/OS Communications Server: IP Messages Volume 1 (EZA)</i>	SC31-8783	This volume contains TCP/IP messages beginning with EZA.
<i>z/OS Communications Server: IP Messages Volume 2 (EZB)</i>	SC31-8784	This volume contains TCP/IP messages beginning with EZB.
<i>z/OS Communications Server: IP Messages Volume 3 (EZY)</i>	SC31-8785	This volume contains TCP/IP messages beginning with EZY.
<i>z/OS Communications Server: IP Messages Volume 4 (EZZ-SNM)</i>	SC31-8786	This volume contains TCP/IP messages beginning with EZZ and SNM.
<i>z/OS Communications Server: IP and SNA Codes</i>	SC31-8791	This document describes codes and other information that appear in z/OS Communications Server messages.

APPC Application Suite:

Title	Number	Description
<i>z/OS Communications Server: APPC Application Suite User's Guide</i>	SC31-8809	This documents the end-user interface (concepts, commands, and messages) for the AFTP, ANAME, and APING facilities of the APPC application suite. Although its primary audience is the end user, administrators and application programmers may also find it useful.
<i>z/OS Communications Server: APPC Application Suite Administration</i>	SC31-8835	This document contains the information that administrators need to configure the APPC application suite and to manage the APING, ANAME, AFTP, and A3270 servers.
<i>z/OS Communications Server: APPC Application Suite Programming</i>	SC31-8834	This document provides the information application programmers need to add the functions of the AFTP and ANAME APIs to their application programs.

Redbooks

The following Redbooks may help you as you implement z/OS Communications Server.

Title	Number
<i>TCP/IP Tutorial and Technical Overview</i>	GG24-3376
<i>SNA and TCP/IP Integration</i>	SG24-5291
<i>IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide: Volume 1: Configuration and Routing</i>	SG24-5227
<i>IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide: Volume 2: UNIX Applications</i>	SG24-5228
<i>IBM Communications Server for OS/390 V2R7 TCP/IP Implementation Guide: Volume 3: MVS Applications</i>	SG24-5229
<i>Secureway Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements</i>	SG24-5631
<i>TCP/IP in a Sysplex</i>	SG24-5235
<i>Managing OS/390 TCP/IP with SNMP</i>	SG24-5866

Title	Number
<i>Security in OS/390-based TCP/IP Networks</i>	SG24-5383
<i>IP Network Design Guide</i>	SG24-2580
<i>Migrating Subarea Networks to an IP Infrastructure</i>	SG24-5957
<i>IBM Communication Controller Migration Guide</i>	SG24-6298

Related information

For information about z/OS products, refer to *z/OS Information Roadmap* (SA22-7500). The Roadmap describes what level of documents are supplied with each release of z/OS Communications Server, as well as describing each z/OS publication.

Relevant RFCs are listed in an appendix of the IP documents. Architectural specifications for the SNA protocol are listed in an appendix of the SNA documents.

The table below lists documents that may be helpful to readers.

Title	Number
<i>z/OS Security Server Firewall Technologies</i>	SC24-5922
<i>S/390: OSA-Express Customer's Guide and Reference</i>	SA22-7403
<i>z/OS JES2 Initialization and Tuning Guide</i>	SA22-7532
<i>z/OS MVS Diagnosis: Procedures</i>	GA22-7587
<i>z/OS MVS Diagnosis: Reference</i>	GA22-7588
<i>z/OS MVS Diagnosis: Tools and Service Aids</i>	GA22-7589
<i>z/OS Security Server LDAP Client Programming</i>	SC24-5924
<i>z/OS Security Server LDAP Server Administration and Use</i>	SC24-5923
<i>Understanding LDAP</i>	SG24-4986
<i>z/OS UNIX System Services Programming: Assembler Callable Services Reference</i>	SA22-7803
<i>z/OS UNIX System Services Command Reference</i>	SA22-7802
<i>z/OS UNIX System Services User's Guide</i>	SA22-7801
<i>z/OS UNIX System Services Planning</i>	GA22-7800
<i>z/OS MVS Using the Subsystem Interface</i>	SA22-7642
<i>z/OS C/C++ Run-Time Library Reference</i>	SA22-7821
<i>z/OS Program Directory</i>	GI10-0670
<i>DNS and BIND</i> , Fourth Edition, O'Reilly and Associates, 2001	ISBN 0-596-00158-4
<i>Routing in the Internet</i> , Christian Huitema (Prentice Hall PTR, 1995)	ISBN 0-13-132192-7
<i>sendmail</i> , Bryan Costales and Eric Allman, O'Reilly and Associates, 1997	ISBN 156592-222-0
<i>TCP/IP Tutorial and Technical Overview</i>	GG24-3376
<i>TCP/IP Illustrated, Volume I: The Protocols</i> , W. Richard Stevens, Addison-Wesley Publishing, 1994	ISBN 0-201-63346-9
<i>TCP/IP Illustrated, Volume II: The Implementation</i> , Gary R. Wright and W. Richard Stevens, Addison-Wesley Publishing, 1995	ISBN 0-201-63354-X
<i>TCP/IP Illustrated, Volume III</i> , W. Richard Stevens, Addison-Wesley Publishing, 1995	ISBN 0-201-63495-3
<i>z/OS System Secure Sockets Layer Programming</i>	SC24-5901

Determining if a publication is current

As needed, IBM updates its publications with new and changed information. For a given publication, updates to the hardcopy and associated BookManager softcopy are usually available at the same time. Sometimes, however, the updates to hardcopy and softcopy are available at different times. The following information describes how to determine if you are looking at the most current copy of a publication:

- At the end of a publication's order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.
- If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.
- To compare softcopy publications, you can check the last two characters of the publication's filename (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk (*) that indicates whether a publication is new or changed.

Summary of changes

Summary of changes for GC31-8782-02 z/OS Version 1 Release 4

This document contains information previously presented in GC31-8782-01, which supports z/OS Version 1 Release 2. The information in this document supports both IPv6 and IPv4. Unless explicitly noted, information describes IPv4 networking protocol. IPv6 support is qualified within the text.

New information:

- Expanded options for diagnosing Sysplex Distributor problems, see “Diagnosing Sysplex Distributor problems” on page 41.
- TCP/IP support for Simple Network Time Protocol (SNTP), see Chapter 33, “Diagnosing Simple Network Time Protocol (SNTP) problems” on page 699.
- Logging FTP server activity, see “Logging FTP server activity” on page 334.
- ND, ROUTEV4 and ROUTEV6 parameters to TCPIP CS TREE subcommand, see “TCPIP CS TREE” on page 250.
- Translation and data conversion support for the FTP server, see “Translation and data conversion support” on page 311.
- Translation and data conversion support for the FTP client, see “Translation and data conversion support” on page 336.
- File tagging support for the FTP client, see “File tagging support” on page 338.
- Steps for resolving the hostname, see “Steps for resolving the hostname” on page 683.
- The following areas contain new information pertaining to IPv6 support:
 - “Controlling the FTP server traces with MODIFY operator command” on page 329.
 - “Filter TCPIP CTRACE by IP address” on page 70.
 - “Formatting event trace records for TCP/IP stacks” on page 71.
 - ROUTE subcommand, “TCPIP CS TREE” on page 250.

An appendix with z/OS product accessibility information has been added.

Changed information:

- 504 replies, see “Server rejects PORT or EPRT command with 504 replies” on page 342.
- Dotted decimal to IP address, see “z/OS UNIX REXEC trace example and explanation” on page 464.
- BIND 9 name server, see “Using the remote name daemon control (rndc) program, BIND 9 name server only” on page 447.

Deleted information:

- UDP parameter from TCPIP CS TREE subcommand.
- VARY PKTTRACE, moved to *z/OS Communications Server: IP System Administrator's Commands*.
- VARY DATTRACE, moved to *z/OS Communications Server: IP System Administrator's Commands*.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R4, you may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our books.

This document supports z/OS.e.

Summary of Changes for GC31-8782-01 z/OS Version 1 Release 2

This document contains information previously presented in GC31-8782-00, which supports z/OS Version 1 Release 1.

New Information:

The following new commands or command parameters:

- z/OS UNIX rndc command (see page 447)
- TCPIP COUNTERS command (see page 178)
- OR'ed trace output for VARY packet trace command (see page 94)
- Policy Agent (see page 532)
- HELP keyword to syntax (see pages 264, 265, 273, 274, 277, 278)
- OMPROUTE (see page 607)

The following enhancements made for FTP:

- ISPF Statistics for PDS Members (see page 323)
- Protecting against bounce attacks (see page 342)
- Controlling FTP Server with MODIFY (see page 329)
- Restarting File Transfers (see page 343)

The following enhancements to tracing:

- Packet Trace Formatting (see page 95)
- Replaceable Static Routes (see pages 217, 225)

Information for BIND-based dynamic domain name server (DNS) version 4.9.3 and version 9 (see pages 441, 443, 443, 449, 451)

The following additions and enhancements for security:

- Resolver Enhancements (see pages 267, 683)
- SMF Recording (see page 217)
- SNMP agent/subagent security (see page 496)
- Policy-Based Intrusion Detection System information (see pages 136, 578)
- Policy (core schema) enhancements (see page 537)
- Enhance Sysplex Distributor in Supporting QoS Policy (see page 261)

Added information for express logon (see page 679)

Changed Information

- The information in the former Appendix A, "Collecting Component Trace Data" has been merged into Chapter 5, "TCP/IP services traces and IPCS support" on page 53.
- EZBIEC07 FFST probe (see page 704)

Deleted Information

- The former Appendix B, "Search Paths" has been deleted because the information is more completely described in the *z/OS Communications Server: IP Configuration Reference*.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Changes for GC31-8782-00 z/OS Version 1 Release 1

This document contains information also presented in *OS/390 V2R10 IBM Communications Server: IP Diagnosis Guide*.

Part 1. General diagnosis information

Chapter 1. Overview of the diagnosis procedure

To diagnose a problem suspected to be caused by z/OS CS, first identify the problem, then determine if it is a problem with TCP/IP. If the problem is TCP/IP-related, gather information about the problem so that you can report the source of the problem to the IBM Software Support Center.

With this information, you can work with IBM® Software Support Center representatives to solve the problem. This document helps you identify the source of the problem.

Figure 1 on page 4 summarizes the procedure to follow to diagnose a problem. The text following the figure provides more information about this procedure.

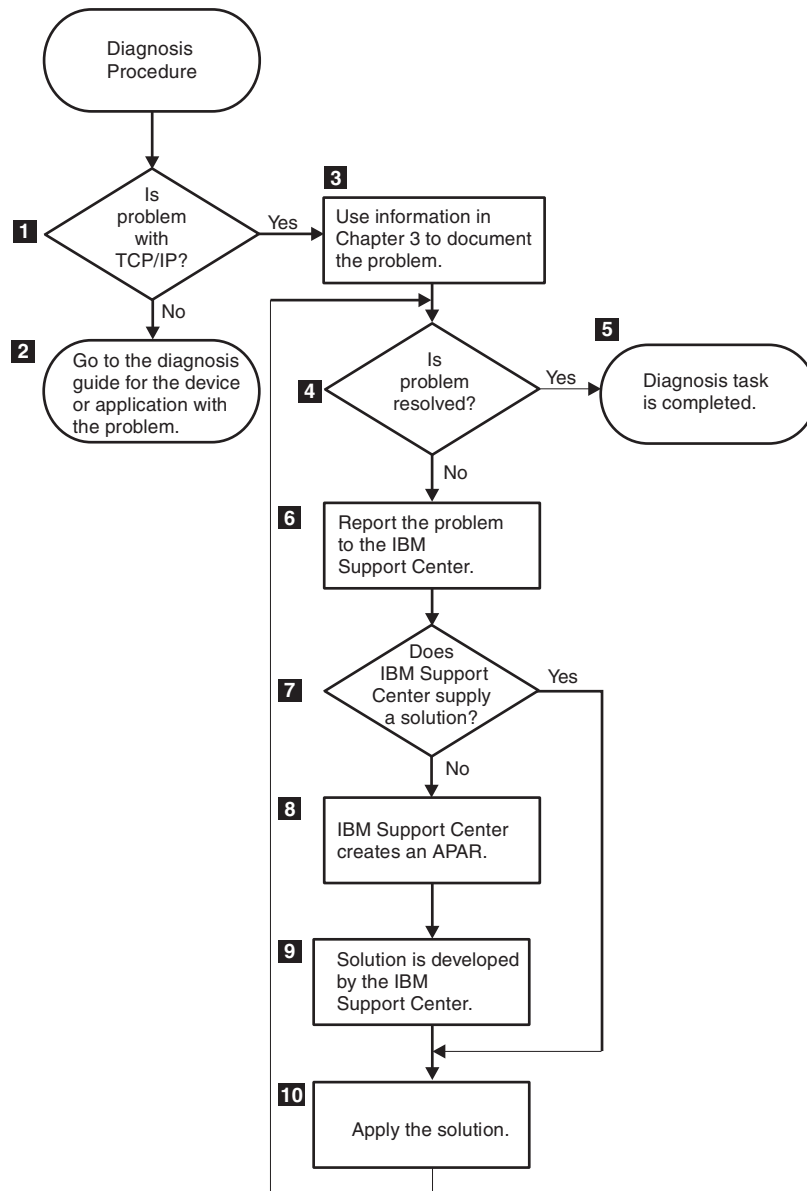


Figure 1. Overview of the diagnosis procedure

1 Determine if the source of the problem is TCP/IP.

Various messages appearing in the console log or in the SYSPRINT or SYSERROR data sets, together with alerts and diagnostic aids, provide information that helps you to find the source of a problem. You should also check syslogd output and be prepared to provide this information to the IBM Software Support Center. If the problem is with TCP/IP, go to Step **3**; otherwise, go to Step **2**.

2 Check appropriate books.

Refer to the diagnosis guide of the hardware device or software application that has the problem.

3 Gather information.

Refer to Chapter 2, “Selecting tools and service aids” on page 7, for a detailed explanation of diagnostic procedures and how to collect information relevant to the problem.

4 Try to solve the problem.

If you cannot solve the problem, go to Step **6**.

5 The diagnosis task is completed.

The problem has been solved.

6 Report the problem to the IBM Software Support Center.

After you have gathered the information that describes the problem, report it to the IBM Software Support Center. If you are an IBMLink® user, you can perform your own RETAIN® searches to help identify problems. Otherwise, a representative uses your information to build keywords to search the RETAIN database for a solution to the problem.

The object of this keyword search using RETAIN is to find a solution by matching the problem with a previously reported problem. When IBM develops a solution for a new problem, it is entered into RETAIN with a description of the problem.

7 Work with IBM Support Center representatives.

If a keyword search matches a previously reported problem, its solution might also correct this problem. If so, go to Step **10**. If a solution to the problem is not found in the RETAIN database, the IBM Software Support Center representatives will continue to work with you to solve the problem. Go to Step **8**.

8 Create an APAR.

If the IBM Software Support Center does not find a solution, they will create an authorized program analysis report (APAR) in the RETAIN database.

9 A solution is developed by the IBM Software Support Center.

Using information supplied in the APAR, IBM Software Support Center representatives determine the cause of the problem and develop a solution for it.

10 Apply the solution.

Apply the corrective procedure supplied by the IBM Software Support Center to correct the problem. Go to Step **4** to verify that the problem is corrected.

Chapter 2. Selecting tools and service aids

This chapter introduces the tools and service aids that z/OS CS provides for diagnosis. As used in this document, the term *tools* includes dumps and traces, while the term *service aids* includes all other facilities provided for diagnosis. For example:

- SVC dump and system trace are tools.
- LOGREC data set and IPCS are service aids.

The following topics are discussed in this chapter:

- “How do I know which tool or service aid to select?” lists problem types and matches them with the appropriate tool or service aid. Use this topic to select the tool or service aid you need for a particular problem.
- “What tools and service aids are available?” on page 12 describes each tool and service aid, including when to use it for diagnosis. Use this topic when you need an overview of tools and service aids, or to find the appropriate time to use a particular tool or service aid.
- “Guidelines for machine-readable documentation” on page 18 describes the guidelines for submitting machine-readable documentation.
- “Submitting documentation electronically” on page 17 describes how to send documentation electronically to IBM using FTP or e-mail.
- “Necessary documentation” on page 19 lists the documentation you need to gather before contacting the IBM Software Support Center.

How do I know which tool or service aid to select?

This section describes the criteria for selecting a tool or service aid, depending on the problem or need:

- Selecting a dump (see Table 1)
- Selecting a TCP/IP services component trace (see Table 2 on page 8)
- Selecting a service aid (see Table 3 on page 11)

The tables show the problem, the corresponding tool or service aid, and the chapter or document that covers it in complete detail. Use these tables to find a tool or service aid quickly.

Refer to “Guidelines for machine-readable documentation” on page 18 for information about submitting dumps and traces to the IBM Software Support Center.

Note: The traces given in this document are only examples. Traces in your environment can differ from these examples because of different options selected.

Table 1. Selecting a dump

What Is the problem?	Type of dump to use
Abnormal end of an authorized program or a problem program.	ABEND dump See “Analyzing abends” on page 21 for detailed information.

Table 1. Selecting a dump (continued)

What Is the problem?	Type of dump to use
TCP/IP server or client address space stops processing or is stopped by the operator because of slowdown or looping condition.	<p>SVC dump</p> <p>The SVC dump is created using the DUMP command.</p> <p>See “Analyzing loops” on page 22 for detailed information.</p>

Table 2. Selecting a trace

What Is the problem?	Type of trace or command to use	Trace output location
<p>Network connectivity</p> <p>See Chapter 4, “Diagnosing network connectivity problems” on page 25 for detailed information.</p>	<p>oping, ARP (onetstat -R)</p> <p>For information on oping, see “Using Ping command” on page 28. For information on onetstat -R, see “onetstat -R” on page 38.</p>	n/a
	<p>Packet trace</p> <p>See Chapter 5, “TCP/IP services traces and IPCS support” on page 53 for detailed information about packet trace.</p>	CTRACE managed data set
<p>Dynamic VIPA or Sysplex Distributor</p> <p>See Chapter 4, “Diagnosing network connectivity problems” on page 25 for detailed information.</p>	Component Trace (SYSTCPIP) XCF option	TCP/IP address space or external writer
<p>TCP/IP socket application</p> <p>See “Socket API traces” on page 74 for detailed information.</p>	Component Trace (SYSTCPIP) SOCKAPI option	TCP/IP address space or external writer
<p>LPR client</p> <p>See “LPR client traces” on page 288 for detailed information.</p>	LPR command with the TRACE option	sysout
<p>LPD server</p> <p>See “LPD server traces” on page 294 for detailed information.</p>	See “LPD server traces” on page 294 for ways to activate traces.	SYSPRINT
<p>z/OS UNIX FTP server</p> <p>See Chapter 8, “Diagnosing File Transfer Protocol (FTP) problems” on page 309 for detailed information.</p>	z/OS UNIX FTP server trace	Server traces appear on the console if syslogd is not started. If it is started, traces appear in the file designated in the syslog.conf file. Refer to the <i>z/OS Communications Server: IP Configuration Guide</i> for detailed information about syslogd.
<p>z/OS UNIX Telnet See Chapter 9, “Diagnosing z/OS UNIX Telnet daemon problems” on page 349, for detailed information.</p>	z/OS UNIX Telnet traces	syslogd

Table 2. Selecting a trace (continued)

What is the problem?	Type of trace or command to use	Trace output location
Telnet See Chapter 10, “Diagnosing Telnet problems” on page 365 for detailed information.	Telnet traces	TCP/IP address space or external writer
SMTP See “SMTP RESOLVER trace” on page 390 for detailed information.	Resolver Trace (see also “Debugging with a resolver directive” on page 447)	Job log output
Popper See Chapter 12, “Diagnosing z/OS UNIX sendmail and popper problems” on page 393 for detailed information.	Popper Messages	syslogd
SNALINK LU0 See Chapter 13, “Diagnosing SNALINK LU0 problems” on page 403 for detailed information.	IP Packet Trace	CTRACE managed data set
	Debug Trace	SNALINK LU0 address space
SNALINK LU6.2 See Chapter 14, “Diagnosing SNALINK LU6.2 problems” on page 417 for detailed information.	TRACE DETAIL ALL	SYSPRINT
	IP Packet Trace	CTRACE managed data set
	TCP/IP Internal Trace	SYSDEBUG
	VTAM® Buffer Trace	GTF managed data set, refer to <i>z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures</i> for detailed information.
Dynamic domain name system (DDNS) See Chapter 15, “Diagnosing name server and dynamic domain name server (DDNS) problems” on page 441 for detailed information.	Error messages	syslogd
	Resolver Trace	Job log output
	TCP/IP component trace	CTRACE managed data set
z/OS UNIX REXEC See Chapter 17, “Diagnosing z/OS UNIX REXEC, REXECD, and RSHD problems” on page 463.	z/OS UNIX REXEC debug trace	syslogd
z/OS UNIX REXECD See Chapter 17, “Diagnosing z/OS UNIX REXEC, REXECD, and RSHD problems” on page 463.	z/OS UNIX REXECD debug trace	syslogd
z/OS UNIX RSHD See Chapter 17, “Diagnosing z/OS UNIX REXEC, REXECD, and RSHD problems” on page 463.	z/OS UNIX RSHD debug trace	syslogd
Network database system (NDB) See Chapter 18, “Diagnosing network database system (NDB) problems” on page 467 for detailed information.	NDB Trace	Job log output

Table 2. Selecting a trace (continued)

What Is the problem?	Type of trace or command to use	Trace output location
X Windows and OSF/Motif See Chapter 19, “Diagnosing X Window System and OSF/Motif problems” on page 483 for detailed information.	XWTRACE and XWTRACEEC (environment variables)	stderr
SNMP See Chapter 20, “Diagnosing Simple Network Management Protocol (SNMP) problems” on page 487 for detailed information.	Manager Traces	Console (osnmp) or SYSPRINT (NetView® SNMP)
	SNMP Agent Traces	syslogd
	TCP/IP Subagent Traces	
	OMPROUTE Subagent Traces	
	SLA Subagent Traces	
	TRAPFWD Traces	
Policy Agent See Chapter 21, “Diagnosing Policy Agent problems” on page 531 for detailed information.	Log file	Refer to the <i>z/OS Communications Server: IP Configuration Guide</i> for detailed information.
RSVP Agent See Chapter 22, “Diagnosing RSVP agent problems” on page 561 for detailed information.	Log file	Refer to the <i>z/OS Communications Server: IP Configuration Guide</i> for detailed information.
Traffic Regulator Management Daemon (TRMD) See Chapter 23, “Diagnosing intrusion detection problems” on page 575 for detailed information.	Log file	syslogd
z/OS UNIX OROUTED See Chapter 24, “Diagnosing OROUTED problems” on page 581 for detailed information.	z/OS UNIX OROUTED trace	Defaults to syslogd. User can use other parameters to send output to the STDOUT DD statement in the OROUTED cataloged procedure. Refer to “Where to send OROUTED trace output” on page 590.
OMPROUTE See Chapter 25, “Diagnosing OMPROUTE problems” on page 601.	Component Trace	CTRACE managed data set
	For detailed information about OMPROUTE Component Trace, see “TCP/IP services component trace for OMPROUTE” on page 615. OMPROUTE Trace For detailed information, see “OMPROUTE traces and debug information” on page 605.	stdout
NCPROUTE See Chapter 26, “Diagnosing NCPROUTE problems” on page 619 for detailed information.	NCPROUTE Traces	SYSPRINT

Table 2. Selecting a trace (continued)

What is the problem?	Type of trace or command to use	Trace output location
X.25 NPSI See Chapter 27, “Diagnosing X.25 NPSI problems” on page 643 for detailed information.	Server activity log	SYSPRINT
IMS See Chapter 28, “Diagnosing IMS problems” on page 653 for detailed information.	IP Packet Trace	CTRACE managed data set
	TCP/IP Internal Trace	CTRACE managed data set
	IMS Trace	Refer to the <i>IMS/ESA Utilities Reference: System</i> for detailed information.
CICS See Chapter 30, “Diagnosing problems with CICS” on page 673 for detailed information.	CICS external trace data set (auxtrace)	Refer to the <i>CICS/ESA 5.2 Problem Determination Guide</i> for detailed information.
	TCP/IP Internal trace	CTRACE managed data set
Express Logon See Chapter 31, “Diagnosing problems with Express Logon” on page 679 for detailed information.	Log file	syslogd
Resolver See Chapter 32, “Diagnosing resolver problems” on page 683 for detailed information.	Trace Resolver	SYSPRINT or stdout
	Resolver Internal trace	CTRACE managed data set

Table 3. Selecting a service aid

What is the problem?	Type of service aid to use
System or hardware problem: need a starting point for diagnosis or diagnosis requires an overview of system and hardware events in chronological order.	LOGREC data set or EREP Refer to <i>z/OS MVS Diagnosis: Tools and Service Aids</i> for detailed information.
Information about the contents of load modules and program objects or a problem with modules on the system.	AMBLIST Refer to <i>z/OS MVS Diagnosis: Tools and Service Aids</i> for detailed information.
Diagnosis requires a trap to catch problem data while a program is running. The DISPLAY TCPIP,STOR command can be used to help set a SLIP trap.	Service Level Indication Processing (SLIP) Refer to <i>z/OS MVS System Commands</i> for detailed information.
Diagnosis requires formatted output of problem data, such as a dump or trace.	IPCS Refer to <i>z/OS MVS IPCS User's Guide</i> for detailed information.

What tools and service aids are available?

This section provides an overview of the tools and service aids in detail. The sections that follow contain a brief description of each tool or service aid, reasons why you would use it, and a reference to the chapter or document that covers the tool or service aid in detail. (Most of the detailed information on tools and service aids is in this document.) A description of tools and service aids are covered in the following sections:

- Dumps (see Table 4)
- Traces (see Table 5 on page 13)
- First Failure Support Technology™ (see “First Failure Support Technology (FFST)” on page 14)
- Display commands (see “Display commands” on page 16)
- System service aids (see Table 6 on page 16)

In the tables that follow, the dumps, traces, or service aids are listed by frequency of use.

Note: The traces given in this document are only examples. Traces in your environment can differ from these examples because of different options selected.

Dumps

Table 4 describes the types of available dumps.

Table 4. Description of dumps

Type of dump	Description
ABEND dumps	<p>Use an ABEND dump when ending an authorized program or a problem program because of an uncorrectable error. These dumps show:</p> <ul style="list-style-type: none">• The virtual storage for the program requesting the dump.• System data associated with the program. <p>The system can produce three types of ABEND dumps— SYSABEND, SYSMDUMP, and SYSUDUMP. Each one dumps different areas. Select the dump that gives the areas needed for diagnosing your problem. The IBM-supplied defaults for each dump are:</p> <ul style="list-style-type: none">• SYSABEND dumps. The largest of the ABEND dumps, containing a summary dump for the failing program plus many other areas useful for analyzing processing in the failing program.• SYSMDUMP dumps. Contains a summary dump for the failing program, plus some system data for the failing task. In most cases, SYSMDUMP dumps are recommended, because they are the only ABEND dumps that are formatted with IPCS.• SYSUDUMP dumps. The smallest of the ABEND dumps, containing only data and areas about the failing program. <p>Reference: Refer to <i>z/OS MVS Diagnosis: Tools and Service Aids</i> for detailed information.</p>

Table 4. Description of dumps (continued)

Type of dump	Description
SVC dumps	<p>SVC dumps can be used in two different ways:</p> <ul style="list-style-type: none"> • Most commonly, a system component requests an SVC dump when an unexpected system error occurs, but the system can continue processing. • An authorized program or the operator can also request an SVC dump when diagnostic data is needed to solve a problem. <p>SVC dumps contain a summary dump, control blocks, and other system code, but the exact areas dumped depend on whether the dump was requested by a macro, command, or SLIP trap. SVC dumps can be analyzed using IPCS.</p> <p>Reference: Refer to <i>z/OS MVS Diagnosis: Tools and Service Aids</i> for detailed information.</p>
FFST™ dumps	<p>FFST dumps fall into two categories: SDUMPs (full dumps) and FFST minidumps (partial dumps). The type of dump produced depends on the characteristics of the probe that produced it.</p> <ul style="list-style-type: none"> • FFST uses the operating system SDUMP macroinstruction to provide a full dump of the address space where the problem occurred. • If the SDUMP option has not been coded for the probe triggering the dump, an FFST minidump is written to the output data set. The probe output data for the TCP/IP minidumps are found in data sets that were allocated when FFST was installed.
Stand-alone dumps	<p>Use a stand-alone dump when:</p> <ul style="list-style-type: none"> • The system stops processing. • The system enters a wait state with or without a wait state code. • The system enters an instruction loop. • The system is processing slowly. <p>These dumps show central storage and some paged-out virtual storage occupied by the system or stand-alone dump program that failed. Stand-alone dumps can be analyzed using IPCS.</p> <p>See “Analyzing loops” on page 22 for detailed information.</p>

Traces

Table 5 describes the types of available traces.

Table 5. Description of traces

Trace	Description
Component trace	<p>Use a component trace when you need trace data to report a client/server component problem to the IBM Software Support Center. Component tracing shows processing between the client and server.</p> <p>Reference: See Chapter 5, “TCP/IP services traces and IPCS support” on page 53 for detailed information.</p>
Data trace	<p>Use a data trace to trace socket data (transforms) into and out of the physical file structure (PFS).</p> <p>Reference: See “Data trace (SYSTCPDA) for TCP/IP stacks” on page 135 for detailed information.</p>

Table 5. Description of traces (continued)

Trace	Description
GTF trace	<p>Use a Generalized Trace Facility (GTF) trace to show system processing through events occurring in the system over time. The installation controls which events are traced.</p> <p>Use GTF when you are familiar enough with the problem to pinpoint the one or two events required to diagnose your system problem. GTF can be run to an external data set.</p> <p>Reference: Refer to <i>z/OS MVS Diagnosis: Tools and Service Aids</i> for detailed information.</p>
Master trace	<p>Use the master trace to show the messages to and from the master console. Master trace is useful because it provides a log of the most recently issued messages. These can be more pertinent to your problem than the messages accompanying the dump itself.</p> <p>You can either accept a dump or write this trace to GTF.</p> <p>Reference: Refer to <i>z/OS MVS Diagnosis: Tools and Service Aids</i> for detailed information.</p>
Packet trace	<p>Use a packet trace to obtain traces of IP packets flowing from and into TCP/IP on a z/OS CS host. The PKTTRACE statement lets you copy IP packets as they enter or leave TCP/IP, and then examine the contents of the copied packets.</p> <p>While the component trace function collects event data about TCP/IP internal processing, packet trace collects data records that flow over the links.</p> <p>Reference: See Chapter 5, “TCP/IP services traces and IPCS support” on page 53 for detailed information.</p>
System trace	<p>Use system trace to see system processing through events occurring in the system over time. System tracing is activated at initialization and, typically, runs continuously. It records many system events, with minimal details about each. The events traced are predetermined, except for branch tracing.</p> <p>You can either take a dump or write this trace to GTF.</p> <p>Reference: Refer to <i>z/OS MVS Diagnosis: Tools and Service Aids</i> for detailed information.</p>
VTAM trace	<p>z/OS CS uses two VTAM® components, CSM and MPC. VTAM traces contain entries for many TCP/IP events, especially I/O and storage requests.</p> <p>Reference: Refer to <i>z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT</i> for detailed information.</p>
z/OS UNIX applications	<p>z/OS UNIX applications send debug and trace output to syslogd. For more information on individual components, such as z/OS UNIX FTP, z/OS UNIX SNMP, or z/OS UNIX OROUTED, refer to those chapters in this manual.</p> <p>Reference: Refer to the <i>z/OS Communications Server: IP Configuration Guide</i> for more detailed information about syslogd.</p>

First Failure Support Technology (FFST)

First Failure Support Technology (FFST) is a licensed program that captures information about a potential problem when it occurs. See Appendix A, “First Failure Support Technology (FFST)” on page 703 for descriptions of the various FFST probes contained in TCP/IP.

Note: For a complete description of FFST commands, refer to the *FFST/MVS FFST/VM Operations Guide*.

When a problem is detected, a software probe is triggered by TCP/IP. FFST then collects information about the problem and generates output to help solve the problem. Based on the options active for the probe, you get a dump and a generic alert. See “Generic alert” for information on generic alerts. You also get the FFST “EPW” message group.

FFST dumps

Each TCP/IP Services FFST probe can trip up to five times in five minutes before it is automatically turned off. Only one of the five dumps will be produced, thereby limiting the number of dumps that you get if a recurring problem triggers a probe.

You get either an SDUMP (full dump) or an FFST minidump (partial dump) depending on the characteristics of the probe that is triggered.

FFST saves the TCP/IP minidump on a dynamically allocated sequential data set. The TCP/IP Services FFST full dump (SDUMP) is saved on SYSLDUMPx data sets. You must specify the volume serial number and the UNIT identification information for this data set. Provide this information to FFST on a DD statement in the FFST installation procedure or in the FFST startup command list installed at system installation. A startup command list contains MVS commands to control FFST.

SDUMP

The SDUMP option is coded in the probe; FFST uses the operating system SDUMP macroinstruction to provide a full dump of the address space where the potential problem occurred.

Formatting an SDUMP

Use the standard IPCS dump formatting and viewing facilities to access the dump. If you use the EPWDMPFM clist to format a full dump, message EPW9561E, NOT A VALID FFST DUMP is issued.

FFST minidump

If the SDUMP option has not been coded for the probe triggering the dump, an FFST minidump is written to the output data set. The probe output data for the TCP/IP minidumps are found in the data sets that were allocated when FFST was installed.

Formatting an FFST minidump

Use the dump formatting CLIST, EPWDMPFM, to format your TCP/IP Services FFST minidump. EPWDMPFM formats your minidump and writes it to a data set you can view online or print using the IEBTPCH utility program.

Generic alert

A software generic alert is built from the symptom record and routed to the NetView program, if installed. The generic alert contains the following:

- Date and time that the probe was triggered
- System name from the CVTSNAME field
- Product name (TCP)
- Component identifier and the release number of the product triggering the probe
- Hardware identification information:
 - Machine type
 - Serial number

- Model number
- Plant code
- Dump data set and volume, if a dump was taken
- Probe statement
- Statement description
- Probe statement severity level

The symptom string

The primary symptom string contains the following data supplied by TCP/IP:

- PIDS/component IP. The TCP/IP component identifier.
- LVLS/level. The TCP/IP specification for the product level.
- PCSS/Probe ID. From the probe that was triggered.
- PCSS/FULL or MINI. Type of dump taken.
- RIDS. Module name from the probe that was triggered.

FFST console

The following is a sample for a console listing for FFST. In this sample, the FFST program console message group “EPW” shows information that a probe has been triggered and that the data is being collected. The EPW0404I message contains the primary symptom string for TCP/IP.

```
EPW0401I FFST390: EVENT DETECTED BY TCP FOR PROBEID EZBXFC05
EPW0406I DUMP DATASET IS: SYSTEM DUMP DATA SET
EPW0402I PRIMARY SYMPTOM STRING FOR PROBEID EZBXFC05 FOLLOWS:
EPW0404I PIDS/5655HAL00 LVLS/50A PCSS/EZBXFC05 PCSS/FULL
EPW0404I RIDS/EZBXFMS0
EPW0701I END OF MESSAGE GROUP
```

Display commands

Display commands can be useful tools and service aids. This section provides a brief description of the DISPLAY TCPIP,,STOR command. For detailed information about this command, refer to the *z/OS Communications Server: IP Configuration Reference*.

DISPLAY TCPIP,,STOR

Use the DISPLAY TCPIP,,STOR command to display TCP/IP active storage or the location and level of a TCP/IP stack module. You can use this command to verify that the load module has the appropriate service level.

System service aids

Table 6 lists the service aids supported by z/OS CS.

Table 6. Description of service aids

Service aid	Description
AMBLIST	<p>Use AMBLIST when you need information about the contents of load modules and program objects or you have a problem related to the modules on your system. AMBLIST is a program that provides extensive data about modules in the system, such as a listing of the load modules, map of the CSECTs in a load module or program object, list of modifications in a CSECT, map of modules in the LPA, and a map of the contents of the DAT-on nucleus.</p> <p>Reference: Refer to the <i>z/OS MVS Diagnosis: Tools and Service Aids</i> for detailed information.</p>

Table 6. Description of service aids (continued)

Service aid	Description
Common storage tracking	<p>Use common storage tracking to collect data about requests to obtain or free storage in CSA, ECSA, SQA, and ESQA. This is useful to identify jobs or address spaces using an excessive amount of common storage or ending without freeing storage.</p> <p>Use Resource Measurement Facility* (RMF*) or the IPCS VERBEXIT VSMDATA subcommand to display common storage tracking data.</p> <p>References:</p> <ul style="list-style-type: none"> • Refer to the <i>z/OS RMF User's Guide</i> for more information about RMF™. • Refer to the <i>z/OS MVS Initialization and Tuning Guide</i> for detailed information about requesting common storage tracking. • Refer to the VSM chapter of the <i>z/OS MVS IPCS User's Guide</i> for information about the IPCS VERBEXIT VSMDATA subcommand.
IPCS	<p>Use IPCS to format and analyze dumps, traces, and other data. IPCS produces reports that can help in diagnosing a problem. Some dumps, such as SNAP, SYSABEND, and SYSUDUMP ABEND dumps, are preformatted and are not formatted using IPCS.</p> <p>Reference: Refer to the <i>z/OS MVS IPCS User's Guide</i> for detailed information.</p>
LOGREC data set	<p>Use the LOGREC data set as a starting point for problem determination. The system records hardware errors, selected software errors, and selected system conditions in the LOGREC data set. LOGREC information gives you an idea of where to look for a problem, supplies symptom data about the failure, and shows the order in which the errors occurred.</p> <p>Reference: Refer to the <i>z/OS MVS Diagnosis: Tools and Service Aids</i> for detailed information.</p>
SLIP traps	<p>Use serviceability level indication processing (SLIP) to set a trap to catch problem data. SLIP can intercept program event recording (PER) or error events. When an event that matches a trap occurs, SLIP performs the problem determination action that you specify:</p> <ul style="list-style-type: none"> • Requesting or suppressing a dump • Writing a trace or a LOGREC data set record • Giving control to a recovery routine • Putting the system in a wait state <p>Reference: Refer to the SLIP command in <i>z/OS MVS System Commands</i> for detailed information.</p>

Submitting documentation electronically

You can send documentation to IBM using File Transfer Protocol (FTP), e-mail, or PUTDOC. If you use FTP, compress all dumps and traces with the TRSMAIN (MVS terse) program, and send the data in BINARY mode. TRSMAIN is prerequisite for PUTDOC.

- To obtain PUTDOC and detailed instruction on its use, follow these steps:
 1. FTP to the Web site at *ftp://service.software.ibm.com*.
 2. Log in using **anonymous** as the user ID and your e-mail address as the password.
 3. Change directories (CD) to */s390/mvs/tools/putdoc/*, where you will find three files: PUTDOC.BIN, PUTDOC.HTML and PUTDOC.SRC.

4. Read the PUTDOC.HTML file for detailed instructions.

Note: TRSMAIN is a prerequisite for PUTDOC.

- To obtain TRSMAIN and detailed instructions on its use, follow these steps:
 1. FTP to the Web site at *ftp://service.software.ibm.com*.
 2. Log in using **anonymous** as the user ID and your e-mail address as the password.
 3. Change directories (CD) to */s390/mvs/tools/packlib/*, where you will find two files: README.TXT and TRSMAIN.
 4. Read the README file for detailed instructions.

If you require any additional directions, call the IBM Support Center.

Guidelines for machine-readable documentation

If, after talking to the IBM Support Center representative about a problem, it is decided that documentation should be submitted to the TCP/IP support team, documentation can be submitted in machine-readable form (that is, on tape). Machine-readable documentation can be handled most efficiently by the IBM Support Center if it conforms to the following guidelines when creating the tape (or tapes).

Note: Trace data created by TCP/IP can contain user IDs, passwords, and other sensitive information. The trace data files should be protected to prevent disclosure. As an example, packet trace of the FTP port 21 used to control FTP sessions contains user IDs and passwords in the CLEAR. However, a customer can use Secure Socket Layer for FTP and for TELNET. The Packet Trace (V TCPIP,,PKTTRACE) command can be RACF® protected.

When preparing machine-readable documentation for submission in an MVS environment, the following guidelines should be followed:

1. Submit dumps and traces on tape.
 - For dumps:

Dump data should not be formatted in any way prior to or during the transfer of the dump to tape.

The DCB parameters of the dump data set should not be changed. The DCB parameters should be:

LRECL=4160, BLKSIZE=4160, RECFM=F (for z/OS CS)
 - For external CTRACE, IP packet trace, and data trace:

CTRACE data should not be formatted in any way prior to or during the transfer to tape. DCB parameters of the CTRACE data set should not be changed.

The IPICS commands COPYDUMP and COPYTRC can also be used. For more information, refer to the *z/OS MVS IPICS Commands*.
 - For GTF traces:

GTF trace data should be moved from the trace data set (which is usually SYS1.TRACE) to tape using IEBGENER only.

The DCB parameters for a GTF trace should be one of the following:

LRECL=4092, BLKSIZE=4096, RECFM=VB
LRECL=4092, BLKSIZE=32760, RECFM=VB

For both traces and dumps, do not reblock the data (that is, do not use a different BLKSIZE value) when moving the information to tape. Only the DCB parameters shown above should be used.

Note: Use of any other utility (IBM or non-IBM) to transfer dump or trace data to tape might result in a processing delay and could result in the APAR being returned to the customer (closed RET), due to the inability of the change team to process the tape.

2. Submit other types of information (such as TCP/IP traces, configuration files, console logs, and so forth) on paper or tape. **Tape is preferable.** If submitted on tape, the data should be written to tape using IEBGENER only. The DCB parameters used when writing this type of data to tape should be the same as the input data set (that is, the same DCB parameters as the source of the data).
3. Tapes that are submitted to the TCP/IP support team may be standard label (SL) or nonlabel (NL). Cartridge (3480) or reel tapes can be used. Each tape should contain an external label to identify the tape and its contents in some way. The problem number or APAR number should appear on the label. If multiple tapes, or multiple files on one tape, are used, a separate explanation should be included itemizing the contents of each tape or file.
4. Include the output from the job used to create each tape with the tapes. It is very important that the IBM Software Support Center have the output from the job that created the tape (not simply the JCL that was used) to verify that the tape was created correctly and that the job completed normally.

Necessary documentation

Before you call the IBM Support Center, have the following information available:

Customer number

The authorization code that allows you to use the IBM Support Center. Your account name, your TCP/IP license number, and other customer identification should also be available.

Problem number

The problem number previously assigned to the problem. If this is your first call about the problem, the support center representative assigns a number to the problem.

Operating system

The operating system that controls the execution of programs (such as MVS/ESA™). Include the MVS or OS/390 release level.

LE runtime library

The release level of the link edit run-time library is also needed if you are compiling user-written applications written in C or C++.

Component ID

A number that is used to search the database for information specific to TCP/IP. If you do not give this number to the support center representative, the amount of time taken to find a solution to your problem increases.

Release number

A number that uniquely identifies each TCP/IP release.

Table 7 on page 20 lists the specific information that you should provide to the IBM Support Center.

Table 7. TCP/IP component name and release level

Component name and release level	System maintenance program	Field maintenance identifier/CLC
z/OS CS V1R4	SMP/E	<p>The following identifiers are associated with this stack:</p> <ul style="list-style-type: none"> • HIP6104 (base) • JIP6129 (HFS) • JIP612K (IP Security Triple DES)

The following are component ID numbers for z/OS CS:

Licensed IBM program

z/OS

Component ID number

5694–A01

A complex problem might require you to talk to several people when you report your problem to the IBM Support Center. Therefore, you should keep all the information that you have gathered readily available. You might want to keep the items that are constantly required, such as the TCP/IP component ID, in a file for easy access.

Chapter 3. Diagnosing abends, loops, and hangs

This chapter contains information about abends, loops, and hangs. More information is given in the individual component chapters in this document.

Analyzing abends

An abend is an abnormal end. Table 8 describes the types of abends that can occur.

Table 8. Types of abends

Type of abend	Description	Where to Find Help
User Abends	User abends are generated by C run-time routines. They usually start with U409x.	Refer to the <i>IBM Open Class Library Reference, Vol. 1</i> and <i>z/OS Communications Server: IP and SNA Codes</i> .
Platform abends	Abend 3C5 and abend 4C5 are internal abends generated by TCP/IP. Note the reason code stored in register 15 and check the IBM database for known problems.	For further assistance, call the IBM Support Center.
System abends	0C4, 0C1, and 878 are system abends.	Refer to the <i>z/OS MVS System Codes</i> .
	0D6/0D4/0C4 abends can occur when an application is removed from VMCF/TNF with the F VMCF/TNF, REMOVE command, or if VMCF is not active when an application or command which requires it is started or issued.	Refer to the <i>z/OS MVS System Codes</i> . Can occur when an application is removed from VMCF/TNF with the F VMCF/TNF, REMOVE command. It can also occur when an application or command, which requires it is started or issued.
CEEDUMPs	LE produces certain types of abends detected for z/OS UNIX applications such as z/OS UNIX Telnet. CEEDUMPs are usually written to the current working directory in the hierarchical file structure.	Refer to the <i>z/OS Language Environment Debugging Guide</i> publication.

A dump is usually produced when TCP/IP or a TCP/IP component address space abends. If an abend occurs and no dump is taken, the dump files or spools might be full or a SYSMDUMP DD statement might not have been specified in the failing procedure. If TCP/IP or a TCP/IP component was not able to complete the dump, you must recreate the abend or wait for it to occur again.

Note: For more information about debugging the abends and the system abends (for example, abends 0C4, 0C1, and 878), refer to the *z/OS MVS Diagnosis: Procedures*.

Analyzing loops

If processing stops or if TCP/IP does not respond to commands, TCP/IP could be in a loop. Some indicators of a loop are:

- Slow response time
- No response at all
- Inordinately high CPU utilization by TCP/IP

If the problem is a loop, use the following procedure to collect documentation:

1. **Get dump output.**

- **Enabled**

Get an SVC dump of TCP/IP or the looping TCP/IP component by issuing the DUMP command from the MVS system console, or press the Program Restart key. Refer to the *z/OS MVS Diagnosis: Tools and Service Aids* for more information about the DUMP command.

Note: Please make sure that the following storage areas are dumped: CSM, TCP/IP, VTAM address space, and any dataspace used by the DLCS, RGN, CSA, LSQA, NUC, PSA, LPA, and TCPIPDS1. TCPIPDS1 is the data space containing the TCP/IP component trace records. For information on dumping this data space, see Chapter 5, “TCP/IP services traces and IPCS support” on page 53.

- **Disabled**

If the loop is disabled, the MVS system console is not available for input. Try the following:

- Use a PSW RESTART to terminate a looping task. This process creates a LOGREC entry with a completion code of X'071'. Use the LOGREC record and the RTM work area to locate the failing module. Depending on the PSW bit 32, the last three bytes (24-bit mode) or four bytes (31-bit mode) contain the address being executed at the time of the dump. Scan the dump output to find the address given in the PSW. For more information on using PSW RESTART, refer to *z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures*.
- Take a stand-alone dump. Refer to *z/OS MVS Diagnosis: Tools and Service Aids* for information about stand-alone dumps.

2. **Get the MVS system console log (SYSLOG), the job log from the started procedure, and the LOGREC output.**

The MVS system console log might contain information, such as error messages, that can help you diagnose the problem. Also, print the LOGREC file.

Use the LOGDATA option to print the in-core LOGREC buffers. Refer to *z/OS MVS Diagnosis: Tools and Service Aids* or *z/OS MVS IPCS Commands* for more information about the LOGDATA option.

Note: The SYSERROR data set might contain additional information to help you diagnose the problem.

3. **Is a message involved?**

Determine whether there are any messages associated with the loop, such as a particular message always preceding the problem, or the same message being issued repeatedly. If so, add the message IDs to your problem documentation.

4. **Examine the trace entries using IPCS.**

By examining all of the trace entries in the system trace table, you might be able to determine whether there is a loop. The most obvious loops would be a module or modules getting continual control of the TCP/IP system.

Use the PSW to determine the names of the modules in the loop. Refer to the *z/OS MVS IPCS User's Guide* for information about using IPCS.

In the output shown in Figure 2, the CLKC entries indicate an enabled loop. The PSW addresses on the CLKCs identify the looping program. Use the WHERE subcommand to locate the responsible program.

02-0029	008E7220	CLKC	078D2600	83A8192C	00001004	00000000
02-0029	008E7220	CLKC	078D2600	83A81934	00001004	00000000
02-0029	008E7220	CLKC	078D2600	83A81930	00001004	00000000
02-0029	008E7220	CLKC	078D2600	83A8192A	00001004	00000000
02-0029	008E7220	CLKC	078D2600	83A81930	00001004	00000000
02-0029	008E7220	CLKC	078D2600	83A81938	00001004	00000000

Figure 2. Example of output from the IPCS SYSTRACE command

Analyzing hangs

If the problem is a hang, use the following procedure to collect documentation:

1. **Determine the extent of the hung state in the operation of the TCP/IP network.**

Determine whether all TCP/IP processing stopped or only processing with respect to a single device, or something in between. Also determine what, if any, recovery action was taken by the operator or user at the time the hang was encountered. Some information about the activity that immediately preceded the hang might be available on the system log or in application program transaction logs.

2. **Does TCP/IP respond to any commands?**

Determine if TCP/IP responds to commands, such as **oping** or **onetstat**. If TCP/IP does not respond to these commands, take an SVC dump of TCP/IP address space and contact the IBM Software Support Center. If TCP/IP does respond to the commands, it is not hung.

3. **Is a particular application (such as z/OS UNIX FTP or a user-written application) hung?**

Take a dump of the OMVS address space, the TCP/IP address space, and the application address space.

Chapter 4. Diagnosing network connectivity problems

Interconnectivity between network hosts encompasses the physical layer or hardware layer, the protocols such as TCP and IP, and the applications that use the services of TCP and IP. Isolating network problems is an essential step in successful implementation of a network application. This chapter introduces commands and techniques you can use to diagnose network connectivity problems.

Diagnostic commands are available for either the z/OS UNIX environment or the TSO environment. The examples in this chapter are for the z/OS UNIX environment. You can substitute the equivalent TSO command, as shown in Table 9. For complete descriptions of these commands, refer to the *z/OS Communications Server: IP User's Guide and Commands*.

Table 9. Diagnostic commands

UNIX command	TSO command	Refer to:
oping	PING	"Using Ping command" on page 28
onetstat	NETSTAT	"Using Netstat command" on page 30
otracer/traceroute	TRACERTE	"Using traceroute" on page 40

Notes:

1. The RESOLVER and NAMESERVER functions, which translate symbolic names to IP addresses, should be avoided when diagnosing network problems. Use the host IP address instead.
2. MVS-style data sets are written in capital letters (for example, *hlq.TCPIP.DATA*). Files names in the hierarchical file system (HFS) are written in lowercase (for example, */etc/hosts*).

For detailed information on internetworking, see Appendix B, "Overview of internetworking" on page 713.

Communicating through the correct stack

If you are running multiple stacks, the first question to ask is whether the application is communicating through the correct stack. To identify the stack an application is using, you can look at the keyword TCPIPjobname in the TCPIP.DATA file. An application can also select a stack using the SETIBMOPT socket API.

You can use the `onetstat -p <tcpname>` command to determine the characteristics of the stack through which the application is currently communicating.

Using the information provided by `onetstat -p <tcpname>`, you can change, if necessary, the *hlq.PROFILE.TCPIP* data set or the application configuration file. Alternatively, the application may need to communicate through another stack.

It is also helpful to understand the search order used by z/OS CS. Refer to the *z/OS Communications Server: IP Configuration Reference* for information about search order.

Problems connecting to the server

Figure 3 shows the steps to take in diagnosing network connectivity problems.

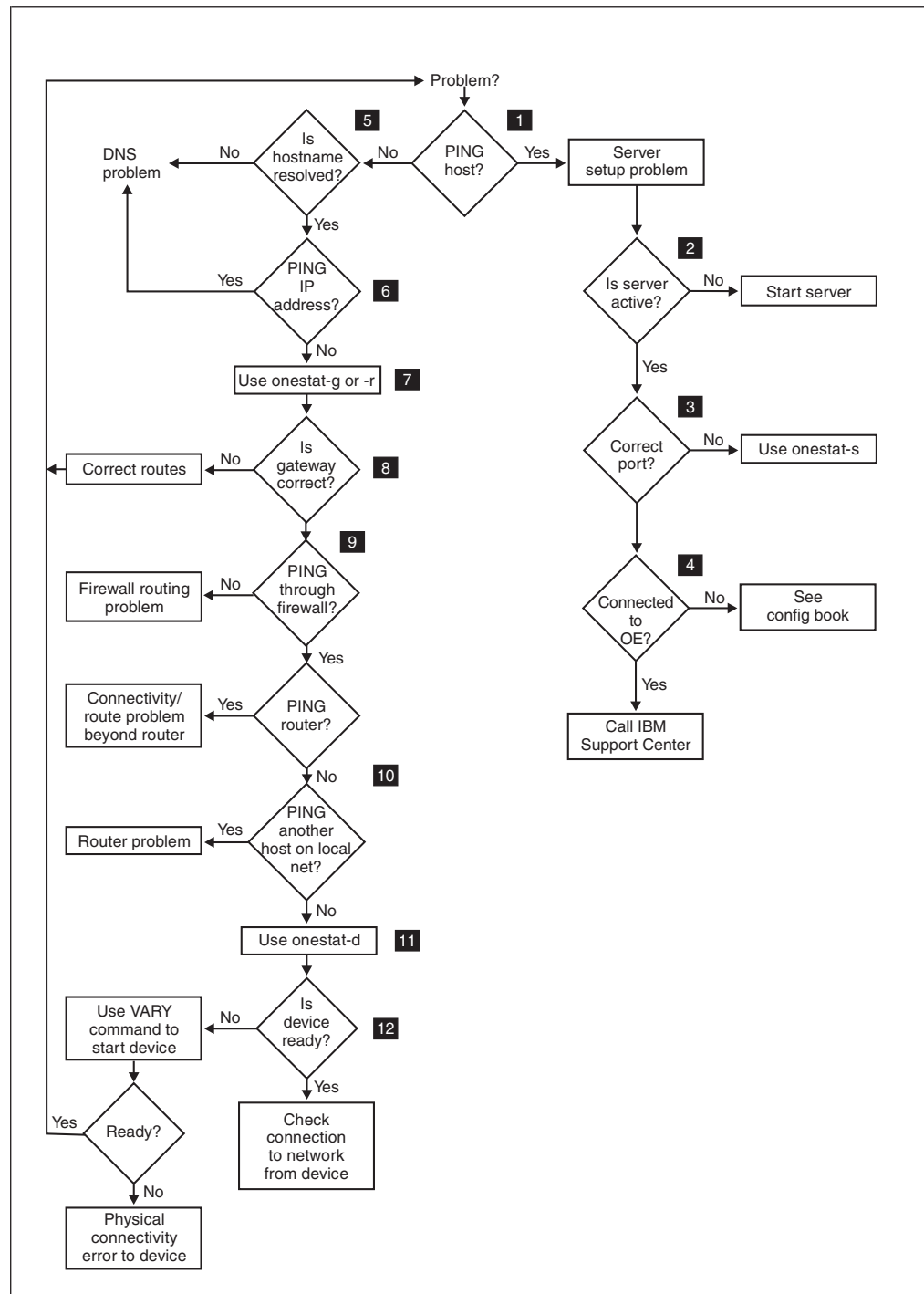


Figure 3. Procedure for diagnosing server connection problems

Following are explanations of the steps in Figure 3.

- 1** Use `oping` loopback to ensure that TCP/IP is running correctly on your host. Then, `oping` one of your home addresses. For basic information about the `oping` command, refer to the *z/OS Communications Server: IP System Administrator's Commands* for details.

Use the `oping` *hostname* command to check connectivity to a remote host. If the host is accessible, you have a server setup problem, continue to Step 3.

Go to Step 5.
- 2** Is the server started? If not, start the server.
- 3** Is the server started on the correct port? Use the `onetstat -s` command to determine which port the server is on. If the server is not on the correct port, configure it correctly.

For basic information about the `onetstat -s` command, refer to the *z/OS Communications Server: IP System Administrator's Commands* for details. For details on server configuration, refer to the *z/OS Communications Server: IP Configuration Reference*.
- 4** Is the server properly connected to z/OS UNIX? If it is not, refer to the *z/OS Communications Server: IP Configuration Reference*. If it is, call the IBM Support Center. For the types of documentation you need to provide, see "Documentation for the IBM Support Center" on page 49.
- 5** Was the *hostname* resolved? If it was not, check the resolver trace. See Chapter 32, "Diagnosing resolver problems" on page 683 for more information about the resolver trace.

If, after checking the trace, you see that the *hostname* was not resolved, you have a name resolution problem. Check the parameters used by the resolver. Verify that the correct TCPIP.DATA files were used and that the parameters are correct. If local host files are used, verify that the correct files were used. In addition, you may want to check other configuration files. For details, refer to the *z/OS Communications Server: IP Configuration Reference*.
- 6** If the *hostname* resolves, use the `oping` command with the IP address of the host to check connectivity. If you get a response, you have a name resolution problem.
- 7** Use the **`onetstat -r`** or the **`onetstat -g`** command to display routes to the network. Verify whether or not TCP/IP has a route to the destination.
- 8** Use the `oping` command to check connectivity to the gateway (router). If you do not get a response, the router may be down or the host may not be connected to the network. If you get a response using the `oping` command, you have a route problem. To resolve it, do the following:

 1. Verify routes to your host.
 2. Check for connectivity or route problems beyond the router.
- 9** Use the `oping` command to check connectivity from the secure to the nonsecure side of the firewall. If you do not get a response, make sure the static routing is set properly on the firewall. To verify static routes, use the **`onetstat -r`** and **`onetstat -h`** commands. The **`onetstat -h`** command shows the addresses for the links (adapters) defined to TCP/IP and **`onetstat -r`** shows information about the static routing. You should have an interface route for each interface and your default route should point to the router on the nonsecure side of the firewall.

If you are able to oping the secure interface on the firewall and the nonsecure interface on the firewall, but are unable to oping the external router, you are probably missing a return route in the external router. The return route in the external router is a static route to your secure network listing the firewall nonsecure interface at the gateway. If you are using unregistered addresses in your secure network, you will not be able to set up a return route.

For additional information about firewall, refer to *z/OS Security Server Firewall Technologies*.

- 10** Use the oping command to check connectivity to another host in the local network.
- 11** Use the onetstat -d command to verify device status. If the device is ready, check the physical connectivity to the network (for example, 3172 or RS/6000® LAN connection).
- 12** Try to start the device using the VARY TCPIP...START, *devname* command. (For information about this command, refer to the *z/OS Communications Server: IP System Administrator's Commands*.) If that is unsuccessful, do the following:
 1. Verify connectivity to the device (channel connector).
 2. Check whether you have a problem at the device itself.

Using Ping command

The packet internet groper (PING) command sends an Internet Control Message Protocol (ICMP) Echo Request to a host, gateway, or router with the expectation of receiving a reply. The **oping** command is the z/OS UNIX version of the TSO PING command. It runs from a UNIX shell.

For a complete description of the PING command, refer to the *z/OS Communications Server: IP System Administrator's Commands*.

Note: The Ping command does not use the sequence number (icmp_seq) to correlate requests with replies. Instead, it uses icmp_id plus an 8-byte TOD time stamp field to correlate requests with replies. The TOD time stamp is the first 8-bytes in the icmp_data field.

- **Use PING or oping loopback to verify the installation of TCP/IP in the z/OS CS environment.**

The oping loopback is essentially an internal software test. This command uses the IPv4 address 127.0.0.1, which is the standard loopback address. An IP packet is not sent to a physical device.

```
oping loopback
CS V1R4: Pinging host loopback (127.0.0.1)
Ping #1 response took 0.007 seconds.
```

For IPv6

```
$ oping::1
CS V1R4: Pinging host::1
Ping #1 response took 0.001 seconds.
```

- **PING or oping a home address to verify the information from the onetstat -h command.**

This is an internal software test. An IP packet is **not** sent to a physical device.

```
ping 9.67.113.58
CS V1R4: Pinging host 9.67.113.58
Ping #1 response took 0.002 seconds.
```

- **PING or oping a host on a directly-attached network to verify the following:**

- The directly-attached network is defined correctly.
- The device is properly connected to the network.
- The device is able to send and receive packets on the network.
- The remote host (193.9.200.2) is able to receive and send packets.

```
oping 9.67.43.101
CS V1R4: Pinging host 9.67.43.101
Ping #1 response took 0.017 seconds.
```

- **PING or oping a host on a remote network to verify the following:**

- The route to the remote network is defined correctly.
- The router is able to forward packets to the remote network.
- The remote host is able to send and receive packets on the network.
- The remote host (9.67.43.126) has a route back to the local host.

```
ping 129.42.19.99
CS V1R4: Pinging host 129.42.19.99
Ping #1 timed out
```

For IPv6

```
oping mvs167
CS V1R4: Pinging host mvs167.tcp.raleigh.ibm.com
at IPv6 address fec9:c2d4::9:67:115:12
Ping #1 timed out
```

Correcting timeout problems

A PING or oping timeout message can occur for many reasons, and various techniques can be used to identify whether the problem is the local z/OS server or a remote host or router. Possible reasons for a timeout are shown in Table 10:

Table 10. Diagnosis of a timeout

Problem	Diagnostic techniques
The device is not transmitting packets to the local network.	Use NETSTAT DEVlinks or onetstat -d to collect information to help you diagnose the problem. (See “onetstat -d” on page 32.)
The remote host is not receiving or transmitting packets on the network.	IPv4-only stack: Use NETSTAT ARp or onetstat -R to display the entry for the remote host. (See “onetstat -R” on page 38.) IPv6-enabled stack: Use NETSTAT ND or onetstat -n to display the entry for the remote host. (See “onetstat -n” on page 40.)
The remote host does not have a route back to the local z/OS server.	Use NETSTAT Gate or onetstat -r on the remote host to make sure it has a route back. (See “onetstat -r” on page 37.)
An intermediate router or gateway is not correctly forwarding IP packets.	Use a packet trace. (See Chapter 5, “TCP/IP services traces and IPCS support” on page 53.)
The ipconfig reassembly timeout value may be set too low.	Refer to the chapter on configuring the TCP/IP address space in <i>z/OS Communications Server: IP Configuration Reference</i> .

Using Netstat command

Use the Netstat command to verify TCP/IP configuration. The following **onetstat** command options are covered in this section. For the equivalent Netstat commands, refer to *z/OS Communications Server: IP System Administrator's Commands*.

- onetstat -h on page 30
- onetstat -d on page 32
- onetstat -g on page 36
- onetstat -r on page 37
- onetstat -R on page 38
- onetstat -n on page 40

Notes:

1. You can run NETSTAT against a TCP/IP stack of the same version. For example, you can run z/OS UNIX onetstat against a z/OS CS stack.
2. The information provided in the output from the onetstat command should be checked against the values in the default configuration data set *hlq.PROFILE.TCPIP*. Refer to the PROFILE DD statement in the TCP/IP started task procedure for the name of the configuration data set.

For details about the **onetstat** command, refer to the *z/OS Communications Server: IP System Administrator's Commands*.

onetstat -h

Use the **onetstat -h** command to verify ADDRESS and LINK values returned by onetstat. Be sure that correct values are coded on the HOME statement in the *hlq.PROFILE.TCPIP* data set.

The following is an example of output on an IPv4-only system:

```
onetstat -h

MVS TCP/IP onetstat CS V1R4      TCP/IP Name: TCPCS      12:34:56
Home address list:
Address      Link      Flg
-----
9.67.113.27  TR1      P
9.67.116.91  CTC1
9.67.116.99  ETH2
9.67.1.8     OSA90LINK1
9.67.116.1   X25LINK
9.67.116.2   CDLCLINK
9.67.116.3   HCHLINK
9.67.116.4   PTPLINK
9.67.116.5   SNA1
9.67.116.6   SNA62
201.2.10.31  VIPLC9020A1F I
127.0.0.1    LOOPBACK
```

The following information is displayed for each entry in the list.

- IP address
- Link name
- Flag

P Primary interface.

I An internally generated VIPA, which is not advertised to routing daemons.

For more information about the home list, refer to *z/OS Communications Server: IP Configuration Reference*.

The following is an example of the display from an IPv6-enabled stack:

```
MVS TCP/IP onetstat CS V1R4      TCP/IP Name: TCPCS      14:23:53
Home address list:
LinkName:  OSAQDIOLINK
  Address:  9.67.115.5
  Flags:    Primary
LinkName:  TR1
  Address:  9.67.113.11
  Flags:
LinkName:  VIPLC9020A1F
  Address:  201.2.10.31
  Flags:    Internal
LinkName:  LOOPBACK
  Address:  127.0.0.1
  Flags:
IntfName:  VIPAV6
  Address:  fec0::a:9:67:115:5
  Type:    Site_Local
  Flags:
  Address:  50c9:c2d4:0:a:9:67:115:5
  Type:    Global
  Flags:    Deprecated
IntfName:  OSAQDIO46
  Address:  fec0::9:67:115:5
  Type:    Site_Local
  Flags:
  Address:  fe80::6:2900:1dc:217c
  Type:    Link_Local
  Flags:    Autoconfigured
  Address:  fec0::6:2900:6dc:217c
  Type:    Site_Local
  Flags:    Autoconfigured
  Address:  50c9:c2d4::6:2900:6dc:217c
  Type:    Global
  Flags:    Autoconfigured
IntfName:  LOOPBACK6
  Address:  ::1
  Type:    Loopback
  Flags:
Unavailable IPv6 Home addresses:
IntfName:  OSAQDIO26
  Address:  fec0::9:67:115:66
  Type:    Site_Local
  Reason:  Duplicate address detection pending start of interface
  Address:  fec0::/64
  Type:    Site_Local
  Reason:  Interface ID not yet known
```

onetstat -F

The onetstat -F command displays dynamic VIPA configuration information:

```
$ onetstat -F
MVS TCP/IP onetstat CS V1R4      TCPIP Name: TCPCS
Dynamic VIPA Information:

VIPA Backup:
  IP Address      Rank
  -----
  201.2.10.13     000100
  201.2.10.14     000100
  201.2.10.21     000080
  201.2.10.22     000080
  201.2.10.31     000060
  201.2.10.32     000040
  201.2.10.33     000060
  201.2.10.34     000040

VIPA Define:
  IP Address      AddressMask      Moveable  SrvMgr
  -----
  201.2.10.11     255.255.255.192  WhenIdle  No
  201.2.10.12     255.255.255.192  Immediate Yes

VIPA Range:
  AddressMask      IP Address      Moveable
  -----
  255.255.255.192  201.2.10.192    Disrupt
  255.255.255.192  201.2.20.192    NonDisr

VIPA Distribute:
  IP Address      Port  XCF Address      SysPt
  -----
  201.2.10.11     00245  201.1.10.85      No
  201.2.10.11     00245  201.1.10.80      No
  201.2.10.11     00245  201.1.10.15      No
  201.2.10.12     04011  ALL              Yes

VIPA Service Manager:
  McastGroup: 224.0.0.1      Port: 04444  Pwd: Yes
```

For details about the onetstat command, refer to the *z/OS Communications Server: IP System Administrator's Commands*.

onetstat -d

Use the onetstat -d command to display the status and associated configuration values for a device and its defined links, as coded in the *hlq.PROFILE.TCPIP* data set.

I The following is an example of output on an IPv4-only system:

```

MVS TCP/IP onetstat CS V1R4          TCP/IP Name: TCPCS          14:23:39
DevName: LOOPBACK                    DevType: LOOPBACK
DevStatus: Ready
LnkName: LOOPBACK                    LnkType: LOOPBACK    LnkStatus: Ready
  NetNum: 0   QueSize: 0
  BytesIn: 24943                      BytesOut: 24943
  ActMtu: 65535
BSD Routing Parameters:
  MTU Size: 00000                      Metric: 00
  DestAddr: 0.0.0.0                    SubnetMask: 0.0.0.0
Multicast Specific:
  Multicast Capability: No

DevName: LCS1                        DevType: LCS          DevNum: 0D00
DevStatus: Ready
LnkName: TR1                          LnkType: TR          LnkStatus: Ready
  NetNum: 0   QueSize: 0
  BytesIn: 9130                      BytesOut: 60392
  MacAddrOrder: Non-Canonical          SrBridgingCapability: Yes
  IpBroadcastCapability: Yes           ArpBroadcastType: All Rings
  MacAddress: 0123456789AB
  ActMtu: 1492
BSD Routing Parameters:
  MTU Size: 02000                      Metric: 100
  DestAddr: 0.0.0.0                    SubnetMask: 255.255.255.128
Packet Trace Setting:
  Protocol: *                          TrRecCnt: 00000006   PckLength: FULL
  SrcPort: *                          DestPort: *
  IpAddr: *                          SubNet: *
Multicast Specific:
  Multicast Capability: Yes
  Group          RefCnt
  -----
  224.0.0.1      0000000001

DevName: OSAQDI04                    DevType: MPCIPA
DevStatus: Ready                     CfgRouter: Pri   ActRouter: Pri
LnkName: OSAQDIOLINK                LnkType: IPAQENET LnkStatus: Ready
  NetNum: 0   QueSize: 0   Speed: 0000000100
  BytesIn: 11476                      BytesOut: 6707
  IpBroadcastCapability: No
  ArpOffload: Yes                      ArpOffloadInfo: Yes
  ActMtu: 1492
BSD Routing Parameters:
  MTU Size: 00000                      Metric: 00
  DestAddr: 0.0.0.0                    SubnetMask: 255.255.255.192
Multicast Specific:
  Multicast Capability: Yes
  Group          RefCnt
  -----
  224.0.0.1      0000000001

DevName: OSATRL90                    DevType: ATM          DevNum: 0000
DevStatus: Not Active
LnkName: OSA90LINK1                 LnkType: ATM          LnkStatus: Not Active
  NetNum: 0   QueSize: 0
  BytesIn: 0                          BytesOut: 0
  ActMtu: Unknown
BSD Routing Parameters:
  MTU Size: 00000                      Metric: 00
  DestAddr: 0.0.0.0                    SubnetMask: 255.0.0.0
ATM Specific:
  ATM portName: OSA90
  ATM PVC Name: STEPH                PVC Status: Not Active

  ATM LIS Name: LIS1
  SubnetValue: 9.67.1.0                SubnetMask: 255.255.255.0
  DefaultMTU: 0000009180              InactvTimeOut: 0000000300
  MinHoldTime: 0000000060             MaxCalls: 0000001000
  CachEntryAge: 0000000900            ATMArpReTry: 0000000002
  ATMArpTimeOut: 0000000003           PeakCellRate: 0000000000
  NumOfSVCs: 0000000000              BearerClass: C

  ATMARPSV Name: ARPSV1

```

```

VcType:      PVC          ATAddrType: NSAP
ATMAddr:
  IAddr:      0.0.0.0
Multicast Specific:
  Multicast Capability: No

DevName: CLAW2          DevType: CLAW      DevNum: 0D10
DevStatus: Ready        CfgPacking: Yes    ActPacking: Packed
LnkName: CLAW2LINK      LnkType: CLAW      LnkStatus: Ready
  NetNum: 0   QueSize: 0
  BytesIn: 0           BytesOut: 0
  ActMtu: 2600
BSD Routing Parameters:
  MTU Size: 00000      Metric: 00
  DestAddr: 0.0.0.0    SubnetMask: 255.255.255.0
Multicast Specific:
  Multicast Capability: Yes

```

The following is an example of output on an IPv6-enabled system:

```

MVS TCP/IP onetstat CS V1R4          TCP/IP Name: TCPCS          14:23:39
DevName: LOOPBACK                    DevType: LOOPBACK
DevStatus: Ready
LnkName: LOOPBACK                    LnkType: LOOPBACK    LnkStatus: Ready
  NetNum: 0   QueSize: 0
  BytesIn: 24943                      BytesOut: 24943
  ActMtu: 65535
BSD Routing Parameters:
  MTU Size: 00000                      Metric: 00
  DestAddr: 0.0.0.0                    SubnetMask: 0.0.0.0
Multicast Specific:
  Multicast Capability: No

IntfName: LOOPBACK6                  IntfType: LOOPBACK6 IntfStatus: Ready
  NetNum: 0   QueSize: 0
  BytesIn: 0                          BytesOut: 0
  ActMtu: 65535
Multicast Specific:
  Multicast Capability: No

DevName: LCS1                        DevType: LCS          DevNum: 0D00
DevStatus: Ready
LnkName: TR1                          LnkType: TR           LnkStatus: Ready
  NetNum: 0   QueSize: 0
  BytesIn: 9130                      BytesOut: 60392
  MacAddrOrder: Non-Canonical        SrBridgingCapability: Yes
  IpBroadcastCapability: Yes         ArpBroadcastType: All Rings
  MacAddress: 0123456789AB
  ActMtu: 1492
BSD Routing Parameters:
  MTU Size: 02000                      Metric: 100
  DestAddr: 0.0.0.0                    SubnetMask: 255.255.255.128
Packet Trace Setting:
  Protocol: *                        TrRecCnt: 00000006   PckLength: FULL
  SrcPort: *                        DestPort: *
  IpAddr: *                          SubNet: *
Multicast Specific:
  Multicast Capability: Yes
  Group                               RefCnt
  ----                               -
  224.0.0.1                          0000000001

DevName: OSAQDI04                    DevType: MPCIPA
DevStatus: Ready
LnkName: OSAQDIOLINK                 LnkType: IPAQENET    LnkStatus: Ready
  NetNum: 0   QueSize: 0   Speed: 0000000100
  BytesIn: 11476                      BytesOut: 6707
  IpBroadcastCapability: No
  CfgRouter: Non                      ActRouter: Non
  ArpOffload: Yes                     ArpOffloadInfo: Yes
  ActMtu: 1492
BSD Routing Parameters:
  MTU Size: 00000                      Metric: 00
  DestAddr: 0.0.0.0                    SubnetMask: 255.255.255.192
Multicast Specific:
  Multicast Capability: Yes
  Group                               RefCnt
  ----                               -
  224.0.0.1                          0000000001

IntfName: OSAQDI046                  IntfType: IPAQENET6 IntfStatus: Ready
  NetNum: 0   QueSize: 0   Speed: 0000000100
  BytesIn: 12655                      BytesOut: 4590
  MacAddress: 000629DC21BC
  SrcVipIntf: VIPAV6
  DupAddrDet: 1
  CfgRouter: Pri                      ActRouter: Pri
  RtrHopLimit: 5
  CfgMtu: 4096                        ActMtu: 1492
Packet Trace Setting:
  Protocol: *                        TrRecCnt: 00000000   PckLength: FULL
  SrcPort: *                        DestPort: *
  IpAddr/PrefixLen: fec0::9:67:115:5/128
Multicast Specific:

```

```

Multicast Capability: Yes
RefCnt      Group
-----
0000000001 ff02::1:ff15:5
0000000001 ff02::1:ffdc:217c
0000000001 ff02::1
0000000001 ff02::1:ff00:2

DevName: VIPAV6          DevType: VIPA
DevStatus: Ready
IntfName: VIPAV6          IntfType: VIPA6      IntfStatus: Ready
NetNum: 0   QueSize: 0
BytesIn: 0                               BytesOut: 0
Packet Trace Setting:
Protocol: *                  TrRecCnt: 00000000 PckLength: FULL
SrcPort: *                  DestPort: *
IpAddr: *                   SubNet: *
Multicast Specific:
Multicast Capability: No

```

Refer to the *z/OS Communications Server: IP System Administrator's Commands* for details on status items displayed.

onetest -g

The **onetest -g** command displays the current routing tables for TCP/IP. In order to establish connectivity to a remote host, the remote host must also have a route back to the z/OS Communications Server. Following is an example of **onetest -g** output:

```

onetest -g

MVS TCP/IP onetest CS V1R4          TCPIP Name: TCPCS          09:37:36
Known gateways:
NetAddress      FirstHop          Link      Pkt Sz Subnet Mask      Subnet Value
-----
Defaultnet      9.67.113.1          TR1       576    <none>
9.67.1.9        <direct>            OSA00LIN 0      HOST
9.0.0.0         <direct>            TR1       576    0.255.255.128  0.67.113.0
9.67.113.43     <direct>            TR1       17914  HOST
127.0.0.1       <direct>            LOOPBACK 65535  HOST
198.11.24.106   198.11.10.210      LCISCH    1024   PMTU_HOST
198.11.25.104   198.11.22.109      LMCH2IT2 26624  HOST
201.2.10.31     <direct>            VIPLC902 65535  HOST

```

Note: New PMTU routes are displayed. Loopback route is displayed. Implicit (HOME list) routes are displayed.

You can use the DETAIL parameter to display gateway information plus MVS specific configured parameters:

```

onetstat -g detail
MVS TCP/IP onetstat CS V1R4          TCPIP Name: TCPCS          09:52:05
Known gateways:
NetAddress      FirstHop      Link      Pkt Sz Subnet Mask      Subnet Value
-----
Defaultnet      9.67.113.1    TR1       576    <none>
Metric: 00000001  Flags: UHS
MVS Specific Configured parameters:
  MaxReTransmitTime: 120.000  MinReTransmitTime: 0.500
  RoundTripGain: 0.125      VarianceGain: 0.250
  VarianceMultiplier: 2.000
9.0.0.0          <direct>      TR1       2000   0.255.255.128  0.67.113.0
Metric: 00000000  Flags: UZ
MVS Specific Configured parameters:
  MaxReTransmitTime: 120.000  MinReTransmitTime: 0.500
  RoundTripGain: 0.125      VarianceGain: 0.250
  VarianceMultiplier: 2.000
9.67.113.27      <direct>      TR1       0       HOST
Metric: 00000000  Flags: UH0
MVS Specific Configured parameters:
  MaxReTransmitTime: 120.000  MinReTransmitTime: 0.500
  RoundTripGain: 0.125      VarianceGain: 0.250
  VarianceMultiplier: 2.000
9.67.116.96      <direct>      LHYDRAF  0       HOST
Metric: 00000000  Flags: UGHR
MVS Specific Configured parameters:
  MaxReTransmitTime: 120.000  MinReTransmitTime: 0.500
  RoundTripGain: 0.125      VarianceGain: 0.250
  VarianceMultiplier: 2.000

```

Refer to the *z/OS Communications Server: IP System Administrator's Commands* for details on status items displayed.

If you note any errors, check *hlq.PROFILE.TCPIP* for the following:

- Make sure no statements were flagged in either the initial profile or in any subsequent VARY TCPIP commands. (For information on the VARY TCPIP command, refer to *z/OS Communications Server: IP System Administrator's Commands*.)
- Make sure the HOME statement has been coded correctly.
- If static routing is provided by way of the GATEWAY statement, make sure the entries in the statement correlate to a valid link name.
- If static routing is provided by way of the GATEWAY statement, make sure there is a GATEWAY or DEFAULTNET entry that correlates to the NETWORK or HOST addresses available on the network.

onetstat -r

The **onetstat -r** displays routing information.

Following is an example of IPv4-only **onetstat -r** output:

```

MVS TCP/IP onetstat CS V1R4          TCPIP Name: TCPCS          14:24:09
Destination      Gateway      Flags      Refcnt  Interface
-----
Default          9.67.115.65  UGS        000002  OSAQDIOLINK
9.67.115.65      0.0.0.0     UHS        000000  OSAQDIOLINK
9.67.115.69      0.0.0.0     UH         000000  OSAQDIOLINK
127.0.0.1        0.0.0.0     UH         000002  LOOPBACK

```

Following is an example of IPv6-enabled **onetstat -r** output:

```

MVS TCP/IP onetstat CS V1R4      TCP/IP Name: TCPCS      14:03:13
IPv4 Destinations
Destination      Gateway      Flags      Refcnt      Interface
-----
Default          9.67.115.65  UGS        000002      OSAQDIOLINK
Metric: 00000001
MVS Specific Configured Parameters:
  MaxReTransmitTime: 120.000  MinReTransmitTime: 0.500
  RoundTripGain:      0.125    VarianceGain:      0.250
  VarianceMultiplier: 2.000
.....
IPv6 Destinations
.....
DestIP:  fec0::206:2aff:fe71:4400/128
Gw:       ::
Intf:     OSAQDI046      Refcnt: 000000
Flgs:     UHS           Pktsize: 1492
Metric: 00000000
MVS Specific Configured Parameters:
  MaxReTransmitTime: 120.000  MinReTransmitTime: 0.500
  RoundTripGain:      0.125    VarianceGain:      0.250
  VarianceMultiplier: 2.000

```

Refer to the *z/OS Communications Server: IP System Administrator's Commands* for details on status items displayed.

The **onetstat -r RSTAT** command displays all of the static routes that are defined as replaceable:

The following is an example of IPv4-only output:

```

MVS TCP/IP onetstat CS V1R4      TCP/IP NAME: TCPCS      17:40:36
IPv4 Destinations
Destination      Gateway      Interface
-----
9.67.1.9         0.0.0.0      OSA00LINK1

```

The following is an example of IPv6-enabled output:

```

MVS TCP/IP onetstat CS V1R4      TCP/IP NAME: TCPCS      17:40:36
IPv4 Destinations
Destination      Gateway      Interface
-----
9.67.1.9/32      0.0.0.0      OSA00LINK1

IPv6 Destinations
DestIP:  fe80::6:2900:1dc:21bc/128
Gw:       ::
Intf:     OSAQDI046

```

onetstat -R

Use the command **onetstat -R <net address>** to query the ARP cache for a given address. Use **onetstat -R ALL** query an entire ARP cache table. Make sure **onetstat -R** displays an ARP entry for the remote hosts.


```
READY  
onetstat -R ALL  
MVS TCP/IP onetstat CS V1R4      TCPIP NAME: TPCPS          12:48:54  
Querying ARP cache for address 201.2.10.32  
Link: SZ_TR1                      IBMTR: 08005A0D97A2  
Route info: 0270  
  
Querying ARP cache for address 201.2.10.31  
Link: SZ_TR1                      IBMTR: 08005A0D97A2  
Route info: 0270  
  
Querying ARP cache for address 9.67.128.1  
Link: IQDIOLNKC0010203  
  
Querying ARP cache for address 9.67.113.43  
Link: SZ_TR1                      IBMTR: 08005A0D97A2  
Route info: 0270  
  
Querying ARP cache for address 9.67.1.8  
Link: OSA90LINK1                 ATMNSAP: 399999999999999999ABCDEFABCD12345678  
  
READY
```

The "Route info" field is the Token Ring Routing Information Field (RIF). Refer to the RIF portion of RFC1042 for detailed information about this field.

Note: The information provided differs depending on the type of device. Refer to the section on devices that support ARP Offload in the *z/OS Communications Server: IP Configuration Guide* for more information on what is supported for each device.

The ARP entry for the host on a remote network contains the destination IP address and the MAC address for the router.

To ensure the host has a route back to the z/OS Communications Server, review the routing tables on the remote host. The route back can be a HOST route or NETWORK route. Intermediate routers must also be configured correctly.

Notes:

1. For more information about **onetest -R**, refer to the *z/OS Communications Server: IP System Administrator's Commands*.
2. The information provided differs depending on the type of device. See *z/OS Communications Server: IP Configuration Guide* for information on what is supported for each device.

onetstat -n

Use **onetstat -n** to display the entry for the remote host.

```
MVS TCP/IP onetstat CS V1R4          TCP/IP Name: TCPCS          14:33:33
Query Neighbor cache for fe80::202:55ff:fe64:2de7
  Intfname: OSAQDI046                Intftype: IPAQENET6
  LinklayerAddr: 000255642DE7        State: Stale
  Type: Host                          AdvDfltRtr: No
Query Neighbor cache for fec0::9:67:114:46
  Intfname: OSAQDI046                Intftype: IPAQENET6
  LinkLayerAddr: 0060CF208827        State: Reachable
  Type: Host                          AdvDfltRtr: No
Query Neighbor cache for fe80::206:2aff:fe71:4400
  IntfName: OSAQDI046                IntfType: IPAQENET6
  LinkLayerAddr: 00062A714400        State: Reachable
  Type: Router                        AdvDfltRtr: Yes
Query Neighbor cache for fe80::206:2aff:fe66:c800
  IntfName: OSAQDI046                IntfType: IPAQENET6
  LinkLayerAddr: 00062A66C800        State: Stale
  Type: Router                        AdvDfltRtr: Yes
```

Using traceroute

The traceroute function displays the route that a packet takes to reach the requested target. Trace starts at the first router and uses a series of UDP probe packets with increasing IP time-to-live (TTL) or hop count values to determine the sequence of routers that must be traversed to reach the target host. The traceroute function can be invoked by either the TSO TRACERTE command or the z/OS UNIX `otracertr` command.

The `packetSize` option lets you increase the IP packet size to see how size affects the route that the traceroute packet takes. It also shows the point of failure if a destination address cannot be reached.

For the complete syntax of the TSO TRACERTE and z/OS UNIX **otracertr** command, refer to the *z/OS Communications Server: IP System Administrator's Commands*.

Examples

- The **otracertr** command issued to a valid destination:

```
otracertr 9.130.40.72
-----
CSV1R4: Traceroute to 9.130.40.72 (9.130.40.72).
Use escape C sequence to interrupt
1 buzz-113.tcp.raleigh.ibm.com (9.67.113.1) 7 ms 56 ms 6 ms
2 b500-503-1.raleigh.ibm.com (9.37.32.1) 25 ms 66 ms 19 ms
3 rtp2wf-atm.raleigh.ibm.com (9.37.1.132) 8 ms 19 ms 23 ms
4 9.32.204.30 (9.32.204.30) 60 ms 16 ms 19 ms
5 9.32.1.69 (9.32.1.69) 49 ms 65 ms 48 ms
6 9.32.44.2 (9.32.44.2) 58 ms * *
7 pok008-1.pok.ibm.com (9.117.1.1) 41 ms 44 ms 58 ms
8 * sqv014-1.endicott.ibm.com (9.130.104.12) 148 ms 93 ms
9 gdlvm7.endicott.ibm.com (9.130.40.72) 91 ms 89 ms 87 ms
```

Following are brief descriptions of some of the items in the preceding example:

- Each line in this example shows a “hop” to a different router.
- An asterisk (*) represents a lost packet.
- The time displayed (7 ms, 56 ms, and so on) is in milliseconds. It shows the round-trip time calculated from when the probe was sent from **otracertr** and when `otracertr` received the ICMP reply. Each time displayed is independent; it is the time it takes for **otracertr** to send a probe and receive a reply.

- The **otracert** command issued to an address of an unreachable host (!H):

```
otracert 1.1.1.1
-----
CSV1R4: TRACEROUTE TO 1.1.1.1 (1.1.1.1).
USE ESCAPE C SEQUENCE TO INTERRUPT
1 buzz-113.tcp.raleigh.ibm.com (9.67.113.1)  5 ms  5 ms  5 ms
2 b500-503-1.raleigh.ibm.com (9.37.32.1)   8 ms  12 ms  8 ms
3 rtp2wf-atm.raleigh.ibm.com (9.37.1.132)  8 ms  12 ms  11 ms
4 9.32.204.30 (9.32.204.30)  30 ms  9 ms  8 ms
5 9.32.204.30 (9.32.204.30)  14 ms !H * *
6 * * *
7 * * 9.32.204.30 (9.32.204.30)  10 ms !H
8 * * *
9 * * *
```

Following are brief descriptions of some of the items in the preceding example:

!H Unreachable host.

- Successful traceroute to an IPv6 destination:

```
Ready;
tracerte linuxipv62.tcp
CS V1R4: Traceroute to LINUXIPV62.TCP.raleigh.ibm.com
at IPv6 address: fec0::1:9:67:114:44
1 fec0::206:2aff:fe66:c800
  (fec0::206:2aff:fe66:c800)  2 ms  3 ms *
2 fec0::1:9:67:114:44
  (fec0::1:9:67:114:44)  2 ms  2 ms  2 ms
```

- Using an unknown IPv6 IP address results in a flag indicating that there is no route to the destination.

```
Ready;
tracerte fec0::1:9:67:114:47
CS V1R4: Traceroute to fec0::1:9:67:114:47
at IPv6 address: fec0::1:9:67:114:47
1 fec0::206:2aff:fe66:c800
  (fec0::206:2aff:fe66:c800)  3 ms !R *  2 ms !R
```

Using SNMP remote PING

Use this command to determine the response time between two remote hosts. For example, from Host A, you can determine the response time (PING) between Hosts B and C, assuming the SNMP agent and TCP/IP subagent are running on Host B. Refer to the *z/OS Communications Server: IP System Administrator's Commands* for details.

Diagnosing Sysplex Distributor problems

Diagnosing Sysplex Distributor problems presents some unique challenges. Because a DVIPA can be associated with multiple stacks in a sysplex, determining where a problem is can be more difficult. Using a combination of **onetstat**, TSO **netstat**, and display sysplex commands provide a clear picture of the sysplex. Refer to the *z/OS Communications Server: IP Configuration Guide* chapter on virtual addressing for an introduction to sysplex distribution with virtual addressing.

Netstat information can be obtained in any of the following ways:

- Using **onetstat** from the z/OS UNIX shell
- Using **netstat** from TSO
- Using the **display netstat** command from the system console

The display netstat command is used in the examples below but any of the three commands can be used.

- For the types of problems where the actual DVIPAs defined on a stack are not what the user expected, confirm the current definitions on a stack (Step 1).
- For Sysplex Distributor workload monitoring, Steps 5 and 6 can be used. If the output from these commands is not what you expected, Step 4 gives you the overall picture of all DVIPA activity in your sysplex.
- If the output from Step 4 reveals an expected target stack not listed for a distributed DVIPA, use Step 2 on the target stack in question. This will help to identify configuration problems on that stack. Note what is required of target stacks. Also verify, with Step 7, that a server application has indeed been activated and bound to the correct port.
- A CTRACE with options XCF, INTERNET, TCP, and VTAMDATA on participating stacks is useful in following the flow of packets into and throughout the sysplex. These can be used to:
 1. Identify the connection being received by the distributing stack.
 2. Determine the stack to which the connection will be forwarded.
 3. Verify the connection being forwarded.
 4. Determine the expected target stack receiving and processing the connection.

Once the connection has been established, subsequent packets can be followed in the same manner. When the connection is terminated, CTRACE records record target stacks, cleaning up the connection and notifying the distributing stack.

The following are the steps to take in diagnosing sysplex problems.

1. Run the display command **d tcpip,tcpics,net,vipadcfg** on the distributing stack to confirm it is configured to distribute the DVIPA and how it is to be distributed. The following report shows that the TCP/IP identified by TCPICS was configured to distribute two DVIPAs. Workload for the first DVIPA, 201.2.10.11 ports 20 and 21, would have been distributed to all stacks in the sysplex including TCPICS itself. Workload for 201.2.10.12, ports 20 and 21, would have been distributed only to the TCP/IP with Dynamic XCF address 193.9.200.2.

```
d tcpip,tcpics,net,vipadcfg
```

```
EZZ2500I NETSTAT CS V1R4 TCPICS 286
DYNAMIC VIPA INFORMATION:
```

VIPA DEFINE:			
IP ADDRESS	ADDRESSMASK	MOVEABLE	SRVMGR
201.2.10.11	255.255.255.240	IMMEDIATE	NO
201.2.10.12	255.255.255.240	IMMEDIATE	NO

VIPA DISTRIBUTE:			
IP ADDRESS	PORT	XCF ADDRESS	SYSPT
201.2.10.11	00020	ALL	NO
201.2.10.11	00021	ALL	NO
201.2.10.12	00020	193.9.200.2	NO
201.2.10.12	00021	193.9.200.2	NO

2. Run the display command **d tcpip,tcpics,net,config** on the distributing stack and all target stacks to confirm that the correct IPCONFIG options have been specified.
 - a. DATAGRAMFWD must be specified on the distributing stack in order for datagrams to be forwarded to target stacks.

- b. SYSPLEXROUTING must be specified on the distributor and all target stacks to get WLM based distribution.
- c. Verify that DYNAMICXCF was specified on the distributor and all target stacks.

Following is the output of this command for the distributing TCP/IP:

```
D TCPIP,TCPCS,N,CONFIG

IP CONFIGURATION TABLE:
FORWARDING: YES      TIMETOLIVE: 00064  RSMTIMEOUT: 00060
FIREWALL:   YES
ARPTIMEOUT: 01200    MAXRMSIZE: 65535  FORMAT:      SHORT
IGREDIRECT: YES      SYSPLXROUT: YES    DOUBLENOP:   NO
STOPCLAWER: NO       SOURCEVIPA: NO    VARSUBNET:   YES
MULTIPATH:  NO       PATHMTUDSC: NO    DEVRTRYDUR: 0000000090
DYNAMICXCF: YES
IPADDR: 193.15.1.1    SUBNET: 255.255.0.0    METRIC: 2
IQDIOROUTE: NO
TCPSTACKSRCVIPA: NO
```

Run the display command D WLM,SYSTEM=MVS004 on the distributing stack and all targets stacks to confirm that WLM is active and in GOAL mode. An example is shown below:

```
D WLM,SYSTEM=MVS004

IWM025I 08.06.58 WLM DISPLAY 610
ACTIVE WORKLOAD MANAGEMENT SERVICE POLICY NAME: CS390SVT
ACTIVATED: 2000/01/11 AT: 19:09:58 BY: xxxxxxxx FROM: yyyyyyyy
DESCRIPTION: WLM POLICY FOR CS/390 SVT Group.
RELATED SERVICE DEFINITION NAME: CS390WLM
INSTALLED: 2000/01/11 AT: 19:09:51 BY: xxxxxxxx FROM: yyyyyyyy
WLM VERSION LEVEL:      LEVEL008
WLM FUNCTIONALITY LEVEL: LEVEL006
WLM CDS FORMAT LEVEL:   FORMAT 3
STRUCTURE SYSZWLM_WORKUNIT STATUS: DISCONNECTED
*SYSNAME* *MODE* *POLICY* *WORKLOAD MANAGEMENT STATUS*
MVS004    GOAL    CS390SVT ACTIVE
```

3. Run the display command **d tcpip,tcpscs,net,vipadyn** on the distributing stack to verify the DVIPA status is ACTIVE and the distribution status is DIST or DIST/DEST. An example is shown below:

```
d tcpip,tcpscs,net,vipadyn

EZZ2500I NETSTAT CS V1R4 TCPCS 570
IP ADDRESS      ADDRESSMASK      STATUS      ORIGINATION      DISTSTAT
201.2.10.11     255.255.255.240 ACTIVE        VIPADEFINE        DIST/DEST
201.2.10.12     255.255.255.240 ACTIVE        VIPADEFINE        DIST
```

Then run display command **d tcpip,tcpscs2,net,vipadyn** on the target stacks to verify it has activated the distributing DVIPA and has it marked as a DEST. In this case TCPCS2 has marked the distributing DVIPAs as DEST and that TCPCS2 is a backup stack for these two DVIPAs (status and origination show backup). An example is shown below:

```
d tcpip,tcpscs2,net,vipadyn

EZZ2500I NETSTAT CS V1R4 TCPCS2 891
IP ADDRESS      ADDRESSMASK      STATUS      ORIGINATION      DISTSTAT
201.2.10.11     255.255.255.240 BACKUP      VIPABACKUP        DEST
201.2.10.12     255.255.255.240 BACKUP      VIPABACKUP        DEST
201.2.10.13     255.255.255.192 ACTIVE        VIPADEFINE
201.2.10.21     255.255.255.192 BACKUP      VIPABACKUP
201.2.10.22     255.255.255.192 BACKUP      VIPABACKUP
```

4. Run the display command **d tcpip,tcps,sysplex,vipad** from any stack in the sysplex to get a global view of how and where DVIPAs are defined within the sysplex and what their status is on each stack. The following display shows:

- Two distributed DVIPAs in the sysplex, 201.2.10.11 and 201.2.10.12
- Which TCP/IPs have been made targets, DIST field = DEST
- The status of all other DVIPAs in this sysplex

```
d tcpip,tcps,sysplex,vipad
```

```
EZZ8260I SYSPLEX CS V1R4 513
VIPA DYNAMIC DISPLAY FROM TCPCS AT MVS004
IPADDR: 201.2.10.11 LINKNAME: VIPLC9020A0B
ORIGIN: VIPADEFINE
TCPNAME MVSNAME STATUS RANK ADDRESS MASK NETWORK PREFIX DIST
-----
TCPCS MVS004 ACTIVE 255.255.255.240 201.2.10.0 BOTH
TCPCS2 MVS004 BACKUP 100 DEST
TCPCS3 MVS004 BACKUP 010 DEST
IPADDR: 201.2.10.12 LINKNAME: VIPLC9020A0C
ORIGIN: VIPADEFINE
TCPNAME MVSNAME STATUS RANK ADDRESS MASK NETWORK PREFIX DIST
-----
TCPCS MVS004 ACTIVE 255.255.255.240 201.2.10.0 DIST
TCPCS2 MVS004 BACKUP 075 DEST
TCPCS3 MVS004 BACKUP 010
IPADDR: 201.2.10.13
ORIGIN: VIPABACKUP
TCPNAME MVSNAME STATUS RANK ADDRESS MASK NETWORK PREFIX DIST
-----
TCPCS2 MVS004 ACTIVE 255.255.255.192 201.2.10.0
TCPCS MVS004 BACKUP 100
TCPCS3 MVS004 BACKUP 010
IPADDR: 201.2.10.21
ORIGIN: VIPABACKUP
TCPNAME MVSNAME STATUS RANK ADDRESS MASK NETWORK PREFIX DIST
-----
TCPCS3 MVS004 ACTIVE 255.255.255.192 201.2.10.0
TCPCS2 MVS004 BACKUP 100
TCPCS MVS004 BACKUP 080
IPADDR: 201.2.10.22
ORIGIN: VIPABACKUP
TCPNAME MVSNAME STATUS RANK ADDRESS MASK NETWORK PREFIX DIST
-----
TCPCS3 MVS004 ACTIVE 255.255.255.192 201.2.10.0
TCPCS MVS004 BACKUP 080
TCPCS2 MVS004 BACKUP 075
15 OF 15 RECORDS DISPLAYED
```

5. Run the display command **d tcpip,tcps,net,vdpt** on the distributing stack to confirm that there are target stacks available with server applications ready. With the keyword **DETAIL** you can also see the current WLM/QOS weight for each service level mapping to a DVIPA/port entry for each Target stack (each DESTXCF ADDR). Refer to the *z/OS Communications Server: IP User's Guide and Commands* for more information.

This display will only show target stacks that are currently up and have joined the sysplex. If there are fewer entries than what showed up in the display command **d tcpip,,net,vipadcfg**, the missing entries may be for target stacks that are not yet up, or are up but did not specify the expected DynamicXCF address.

```
D TCPIP,TCPCS,NETSTAT,VDPT,DETAIL
EZZ2500I NETSTAT CS V1R4 TCPCS 743
DYNAMIC VIPA DISTRIBUTION PORT TABLE:
DEST IPADDR DPORT DESTXCF ADDR RDY TOTALCONN WLM
197.11.200.1 00080 199.11.87.104 001 0000410485 01
```

	QOSPLCACT: *DEFAULT*	W/Q: 01
	197.11.200.1 00080 199.11.87.106 001 0000393807 04	
	QOSPLCACT: *DEFAULT*	W/Q: 04
	QOSPLCACT: PAPRD-SD-7-INTR-DONTCARE	W/Q: 02
	QOSPLCACT: PAPRD-SD-7-INTR-SLOW	W/Q: 01
	QOSPLCACT: PAPRD-SD-7-INTR-SPECIAL	W/Q: 04

Examine the READY (RDY) count fields. If the count is 0, then no server application has been activated or no server is LISTENing on DPORT on the target stack identified by the DESTXCF ADDR. For servers that use more than one port, the RDY value reflects the port the LISTEN is performed on. For example, for FTP, the control connection port (port 21) is where you would expect to see a RDY count greater than 0. If the ready count is not what you expect, proceed to Step 7 to verify whether any server is LISTENing on the DPORT on the target stack.

Check the TotalConn count to see the distribution history. This is a cumulative count of the number of connections that have been forwarded by the distributing stack to each target stack.

If the connections are not being distributed to the target stacks as expected and either the WLM field or the W/Q field contains 00, then consider the following:

- If using WLM to distribute connections based upon the workload of the target stacks, verify that ALL participating stacks (the distributor and all targets) have SYSPLEXROUTING specified (see Step 2 above for instructions for verifying this). Also, verify that WLM is configured and active in GOAL mode on all participating stacks (see Step 2 above).
 - If the WLM configuration appears correct, consider whether the unexpected distribution results may be due to the current workload on the target stacks.
 - If the unexpected distribution results have not yet been explained and Sysplex Distributor Performance Policies have been defined using Policy Agent, consider whether the distribution may be due to network performance (TCP retransmissions and timeouts).
 - If Sysplex Distributor Routing Policies have been defined using Policy Agent, consider whether the definition of that policy is affecting the connection distribution. After determining which connections are not being distributed correctly, run **d tcpip,tcps,net,vcrt,detail** (see below) to determine the policy action that each connection maps to. Look at the QoS weights for those policy actions in the **vdpt** detail display to see if they are unusually low. The Policy Agent log on the target stack can display for each DVIPA/Port the QoS service level fractions used to modify the QoS weight. It can also display the calculations that caused a QoS fraction to be set abnormally high (such as connection limit exceeded or throughput exceeded). See "Diagnosing Policy Agent problems" on page 534 for more information.
6. Run the display command **d tcpip,tcps,net,vcrt** on the distributing stack to see if there are any active connections being routed by the distributor. If you run the command with the keyword DETAIL (**d tcpip,tcps,net,vcrt,detail**) you can see the policy rule and policy action that each connection maps to.

If this table is empty, then connection requests may not be getting to the distributor. Check for a routing problem from the client to the distributor.

If you see expected entries in the table, note the Dynamic XCF address and proceed to Step 7.

```
d tcpip,tcps,net,vcrt
```

```
EZZ2500I NETSTAT CS V1R4 TCPCS 363
```


DYNAMIC VIPA CONNECTION ROUTING TABLE:				
DEST IPADDR	DPORT	SRC IPADDR	SPORT	DESTXCF ADDR
201.2.10.11	00021	193.9.200.5	01029	193.9.200.1
201.2.10.11	00021	193.9.200.8	01050	193.9.200.2
201.2.10.11	00021	193.9.200.11	01079	193.9.200.3
201.2.10.12	00021	193.9.200.9	01030	193.9.200.2

```

D TCPIP,TCPCS,NETSTAT,VCRT,DETAIL
EZZ2500I NETSTAT CS V1R4 TCPCS 779
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST IPADDR      DPORT  SRC IPADDR      SPORT  DESTXCF ADDR
201.2.10.11      00021  193.9.200.1     01060  193.1.1.18
POLICYRULE:      FTPD1
POLICYACTION:     PAPRD-SD-7-INTR-SPECIAL
203.1.10.18      08000  193.10.1.118    01080  193.1.1.108
POLICYRULE:      PRULE-TCP-HIGH
POLICYACTION:     PACTION-TCP-HIGH

```

- Go to the target stacks represented by the DESTXCF ADDR field in the VCRT or VDPT display and run the display command **d tcpip,tcps3,net,allconn,ipa=201.2.10.11** to see the connections on the target stack.

```
d tcpip,tcps3,net,allconn,ipa=201.2.10.11
```

```

01.45.56 EZZ2500I NETSTAT CS V1R4 TCPCS3 737
USER ID  CONN      LOCAL SOCKET      FOREIGN SOCKET      STATE
FTPD1    00000034      201.2.10.11..21   193.9.200.11..1079  ESTBLSH

```

Note: For a variety of reasons, the VCRT and ALLCONN displays may not match exactly. For example, with short-lived connections such as Web connections, an entry may show up in one display but be gone by the time the second display is run. Also, the distributing stack places an entry into the Dynamic VIPA Connection Routing Table when it first forwards a connection request. A busy server may reject these connection requests and therefore cause a temporary mismatch in the two displays.

Diagnosing Sysplex-wide dynamic source VIPAs for TCP connections problems

- Run the netstat config command on the target stacks for which IPCONFIG TCPSTACKSOURCEVIPA was configured to verify the sysplex wide dynamic source VIPA was configured as expected. For example:

```

IP CONFIGURATION TABLE:
FORWARDING: YES      TIMETOLIVE: 00064  RSMTIMEOUT: 00060
FIREWALL: NO
ARPTIMEOUT: 01200    MAXRMSIZE: 65535   FORMAT: SHORT
IGREDIRECT: NO       SYSPLXROUT: YES    DOUBLENOP: NO
STOPCLAWER: NO       SOURCEVIPA: YES    VARSUBNET: YES
MULTIPATH: NO        PATHMTUDSC: NO     DEVRTRYDUR: 0000000090
DYNAMICXCF: YES
IPADDR: 193.9.200.3   SUBNET: 255.255.255.248  METRIC: 01
IQDIOROUTE: NO
TCPSTACKSRCVIPA: 201.1.10.10

```

- Create an outbound connection without having issued a bind (implicit bind) first. Use netstat ALLCONN to confirm that the correct source IP address was used.

Diagnosing SYSPLEXEXPORTS Problems

- Run the netstat vipadcfg command on the distributor stack to confirm that SYSPLEXEXPORTS was specified for all distributed Dynamic VIPAs that you expected.


```

NETSTAT VIPADCFG
MVS TCP/IP NETSTAT CS V1R4          TCP/IP Name: TCPCS          10:30:09
Dynamic VIPA Information:
VIPA Backup:
  IP Address      Rank
  -----
  201.2.10.13     000100
  201.2.10.21     000080
  201.2.10.22     000080
VIPA Define:
  IP Address      AddressMask      Moveable  SrvMgr
  -----
  201.2.10.11     255.255.255.240 Immediate Yes
  201.2.10.12     255.255.255.240 WhenIdle  No
VIPA Range:
  AddressMask      IP Address      Moveable
  -----
  255.255.255.192 201.2.10.192 NonDisr
  255.255.255.192 201.2.20.192 Disrupt
VIPA Distribute:
  IP Address      Port    XCF Address      SysPt
  -----
  201.2.10.11     00245   201.1.10.85      Yes
  201.2.10.11     00245   201.1.10.80      Yes
  201.2.10.11     00245   201.1.10.15      Yes
  201.2.10.12     04011   ALL               No
VIPA Service Manager:
McastGroup: 224.0.0.1      Port: 01472  Pwd: Yes

```

In the display above, the distributed Dynamic VIPA 201.2.10.11 was enabled with SYSPLEXPORTS(SysPt is Yes) while 201.2.10.12 was not (SysPt is No).

2. Verify the system log that the following message was issued:

```
IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORT
```

If this message was not issued and netstat vipadcfg shows that SYSPLEXPORTS was specified, see SNA Network Implementation Guide for more information about defining EZBEPORT with the coupling facility.

3. Create an outbound connection with the source IP address of the SYSPLEXPORTS DRVIPA. Do the following to verify SYSPLEXPORTS is working correctly:

- a. Issue netstat ALLCONN to verify the connection on the target stack.
- b. Issue netstat VCRT to confirm the distributing stack knows about the connection.
- c. Issue the VTAM display net stats command to confirm that the coupling facility is managing ports for this DRVIPA. For example:

```

d net,stats,type=cfs,strname=ezebport,dvipa=9.43.242.3,scope=all
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = STATS,TYPE=CFS
NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORT
STRUCTURE TYPE = LIST
LIST HEADERS = 1024 - LOCK HEADERS = 1024
STORAGE ELEMENT SIZE = 256

```

	CURRENT	MAXIMUM	PERCENT
STRUCTURE SIZE	26K	5026K	0
STORAGE ELEMENTS	96	19136	0
LIST ENTRIES	4	598	0

LIST DVIPA	SYSNAME	TCPNAME	# ASSIGNED PORTS
1	9.43.242.3		2
	VIC002	TCPCS4	1

PORTS: 1025
VIC002 TCPCS3
PORTS: 1024

1

END

Diagnosing Sysplex Wide Security Association (SWSA) problems

Please refer to the *z/OS Security Server Firewall Technologies* guide for details on setting up and diagnosing your security associations. The information here is for aiding in diagnosing Sysplex Wide Security Association (SWSA) specifically.

Basic guidelines: Ensure you have consistent IPSec policies on all participating systems:

- Distributing stacks, target stacks and backup stacks.
 - Certificates identifying hosts must be available on all hosts. That means that RACF (the repository for the IKE certificates) must be shareable between processors in the Sysplex.
1. To take advantage of SWSA you must code the DVIPSEC option on your distributing and backup/takeover stacks. Run the netstat config command on the stacks in question to confirm that FIREWALL and DVIPSEC were specified on the IPCONFIG statement.

Netstat Config MVS TCP/IP onetstat CS V1R4 TCPIP Name: TCPCS 18:14:48

IP Configuration Table:

Forwarding: Yes TimeToLive: 00064 RsmTimeOut: 00060
FireWall: Yes DVIPSec: Yes

2. For SWSA functions to work correctly the stacks involved must be connected to the EZBDVIPA coupling facility structure. Verify from the System Log for the distributing and target stacks (for Sysplex distribution of IPSec workload) and the primary and backup stacks (for Dynamic Tunnel recovery) that the following message was issued:

IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBDVIPA

If this message was not issued and netstat config shows that DVIPSEC was specified, see the *z/OS Communications Server: SNA Network Implementation Guide*, for information about setting up the sysplex environment for VTAM function and defining EZBDVIPA with the coupling facility.

3. For Sysplex distribution of IPSec traffic, the target stacks must have a copy of the Dynamic Tunnel, called a shadow tunnel, that matches the dynamic tunnel on the distributing stack. Enter the following firewall services command on the distributing stack:

fwdynconns cmd=listactive 2 203.15.1.1 203.110.1.1 inbound/outbound remote

Then enter the following command on the target stacks:

fwdynconns cmd=listshadow 2 203.15.1.1 203.110.1.1 inbound only shadow

4. Information about the dynamic tunnels used in SWSA is kept in the coupling facility structure in the event a recovery of the tunnel is necessary. For example, the Dynamic VIPA the tunnel was created for has been taken over by another stack in the sysplex. For IPSec connections to still function using that Dynamic VIPA the tunnel would have to be recovered by the same stack that took over the Dynamic VIPA. To confirm that the coupling facility has the information about the tunnels in the event a recovery is necessary, enter the following VTAM command:

d net,stats,type=cfs,strname=ezbdivipa,dvipa=203.15.1.1
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = STATS,TYPE=CFS
NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBDVIPA

```

STRUCTURE TYPE = LIST
LIST HEADERS = 1024 - LOCK HEADERS = 0
STORAGE ELEMENT SIZE = 256
-----
CURRENT MAXIMUM PERCENT
STRUCTURE SIZE 17K 6896K 0
STORAGE ELEMENTS 32 14839 0
LIST ENTRIES 4 1485 0
-----
LIST DVIPA SYSNAME TCPNAME #ENTRIES TGCCOUNT SEQNUMBER
1 203.15.1.1
VIC015 TCPCS 2 1

```

The list entry for the Dynamic VIPA (list 1 above) shows for which system and which stack the coupling facility is maintaining information about the tunnel.

5. For Sysplex distribution of IPsec traffic, the dynamic tunnel's replay count (sequence number) is maintained in the EZBDVIPA coupling facility structure. This replay count is shared by the distributing stack's dynamic tunnel and all the target stack's shadow tunnels. Enter the following VTAM command to confirm the coupling facility is managing the replay count:

```

d net,stats,type=cfs,strname=ezbdivipa,scope=all,list=all
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = STATS,TYPE=CFS
NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBDVIPA
STRUCTURE TYPE = LIST
LIST HEADERS = 1024 - LOCK HEADERS = 0
STORAGE ELEMENT SIZE = 256
-----
CURRENT MAXIMUM PERCENT
STRUCTURE SIZE 17K 6896K 0
STORAGE ELEMENTS 32 14839 0
LIST ENTRIES 4 1485 0
-----
LIST DVIPA SYSNAME TCPNAME #ENTRIES TGCCOUNT SEQNUMBER
1 203.15.1.1
VIC015 TCPCS 2 1
3 203.15.1.1
VIC015 TCPCS 1 1 END

```

The list entry for the dynamic VIPA that has a value in the SEQNUMBER column, list 3 above, confirms this tunnel's replay count is being managed by the coupling facility.

Documentation for the IBM Support Center

Persistent error conditions, in most cases, indicate an installation or configuration problem. Contact the local IBM branch office for installation assistance. If a software defect is suspected, collect the following information prior to contacting the IBM Support Center:

- PROFILE.TCPIP
- TCPIP.DATA
- Output from **onetstat** commands
- Output from **onetstat -r** or **oping** traces
- Network diagram or layout
- Error messages received (refer to *z/OS Communications Server: IP Messages Volume 1 (EZA)* for information about messages)
- OMPROUTE component trace in the CTRACE managed data set and OMPROUTE trace in stdout

Part 2. Traces and control blocks

Chapter 5. TCP/IP services traces and IPCS support

This chapter describes selected procedures for TCP/IP Services component trace, packet trace, and Socket API trace.

Component trace

You typically use component trace when re-creating a problem. Component Trace performs the following functions:

- Captures trace requests.
- Adds trace records to an internal buffer.
- Writes the internal buffer to an external writer, if requested.
- Formats the trace records using the Interactive Problem Control System (IPCS) subcommand CTRACE.
- Provides a descriptor at the beginning of a trace record that specifies the address and length of each data area. Each data area in the trace record is dumped separately.
- Provides an optional identifier for the connection (UDP, TCP, and so on) as part of each record.

Note: Trace data created by TCP/IP can contain user IDs, passwords, and other sensitive information. The trace data files should be protected to prevent disclosure. As an example, packet trace of the FTP port 21 used to control FTP sessions contains user IDs and passwords in the CLEAR. However, a customer can use Secure Socket Layer for FTP and for TELNET. The Packet Trace (V TCPIP,,PKTTRACE) command can be RACF protected.

For detailed information, refer to the following books:

- *z/OS MVS Diagnosis: Tools and Service Aids* for information about Component Trace Procedures.
- *z/OS MVS Initialization and Tuning Reference* for information about the Component Trace SYS1.PARMLIB member.
- *z/OS MVS System Commands* for information about commands.
- *z/OS MVS Programming: Authorized Assembler Services Guide* for procedures and return codes for component trace macros.

Modifying options with the TRACE CT command

After initialization, you must use the TRACE CT command to change the component trace options. All options except for the size of the TCP/IP component trace buffer can be changed using this command. Modifying options with the TRACE CT command can be done with or without the PARMLIB member.

With PARMLIB member

To change component trace options using a PARMLIB member, create a new SYS1.PARMLIB member and specify the component member on the PARM= keyword of the TRACE CT command. Use the following syntax:

```
TRACE CT,ON,COMP=component_name,SUB=(procedure_jobname)
,PARM=parmlib_member
```

Following are descriptions of the parameters:

COMP Indicates the component name:

SYSTCPDA

TCP/IP packet trace. There is no parmlib member. Options are specified by the VARY TCPIP, ,PKTTRACE command. (see “Packet trace (SYSTCPDA) for TCP/IP stacks” on page 92).

SYSTCPIP

TCP/IP event trace, parmlib = CTIEZBxx, where xx is any 2 alphanumeric characters (see “Event trace (SYSTCPIP) for TCP/IP stacks” on page 62).

SYSTCPIS

TCP/IP intrusion detection service, parmlib = CTIIDSxx (see “Intrusion Detection Services trace (SYSTCPIS)” on page 136).

SYSTCPRE

Resolver, parmlib = CTIRESxx, (see Chapter 32, “Diagnosing resolver problems” on page 683).

SYSTCPRT

OMPROUTE, parmlib = CTIORA00 (see “TCP/IP services component trace for OMPROUTE” on page 615).

SUB Indicates the started procedure name for TCP/IP, the OMPROUTE application, or the RESOLVER started task name for which the trace is run. If you use the **S procname.jobname** method of starting TCP/IP or OMPROUTE, the value specified for *jobname* must be the same as that for the SUB parameter. There can be as many as eight TCP/IP sessions active in one system. Only one OMPROUTE application can be active on each TCP/IP stack.

PARM Identifies the parmlib member containing the trace options (see “COMP” on page 53). All options except the size of trace buffers can be respecified. The size cannot be changed during the execution of TCP/IP, OMPROUTE, or the Resolver. If a different size is required, you must stop TCP/IP, OMPROUTE, or the Resolver, and then restart it after modifying the parmlib member.

If the incorrect parmlib member is specified, one of the following messages may be issued:

- An incorrect CTIEZBxx member is specified on the TRACE CT,ON command:

```
IEE538I CTIEZBxx MEMBER NOT FOUND IN SYS1.PARMLIB
ITT010I COMPONENT TRACE PROCESSING FAILED FOR PARMLIB MEMBER=CTIEZBxx:
PARMLIB MEMBER NOT FOUND.
```

- An incorrect CTIEZBxx member is specified on the CTRACE() keyword of the EXEC statement of the TCP/IP started procedure:

```
IEE538I CTIEZBxx MEMBER NOT FOUND IN SYS1.PARMLIB
```

- An incorrect CTIORAxx member is specified on the TRACE CT,ON command:

```
IEE538I CTIORAxx MEMBER NOT FOUND in SYS1.PARMLIB
ITT010I COMPONENT TRACE PROCESSING FAILED FOR PARMLIB MEMBER=CTIORAxx:
PARMLIB MEMBER NOT FOUND
```

Without PARMLIB member

To change component trace options without using a PARMLIB member, issue the TRACE CT command without the PARM= parameter and specify the options on the reply. Though the SYSTCPDA component for packet or data trace does not have a parmlib member, SYSTCPDA can be used on the trace command without the PARMLIB member. Use the following syntax:

```
TRACE CT,ON,COMP=component_name,SUB=(procedure_jobname)
```


After issuing the TRACE CT command, you are prompted to specify the trace options. Respond using the following syntax:

```
Reply nn
[,ASID=(asid-list)]
[,JOBNAME=(jobname-list)]
[,OPTIONS=(name [name] ...)]
[,WTR={membername | DISCONNECT}]
[,CONT | END]
```

Note: ASID and JOBNAME are not valid for OMPROUTE.

Reply *nn* Specifies the identification number (in the range 0-9999) in the prompting message. For example, if the response is

```
06 ITT066A SPECIFY OPERAND(S) FOR TRACE CT COMMAND
```

You might reply

```
r 06,WTR=PTTCP,END
```

ASID The ASID (address space identifiers) of the client whose TCP/IP requests are to be traced.

JOBNAME The job name of the client whose TCP/IP requests are to be traced. The job name may be:

- The job name associated with a client application.
- The SNA LU associated with a TELNET session.

Note: The JOBNAME parm should not be used with the TELNET ctract option.

- The FTP user ID associated with a FTP data connection.

OPTIONS Options valid for use with SYSTCPIP are listed in Chapter 5, “TCP/IP services traces and IPCS support” on page 53; options valid for use with OMPROUTE are listed in Chapter 25, “Diagnosing OMPROUTE problems” on page 601; and options for SYSTCPRE (the Resolver component) are listed in Chapter 32, “Diagnosing resolver problems” on page 683.

membername The member containing the source JCL that invokes the external writer. The *membername* in the WTR parameter must match the *membername* in a previous TRACE CT,WTRSTART command. (See “Obtaining component trace data with an external writer” on page 56.)

WTR=DISCONNECT

Disconnects the component trace external writer and the trace. You must also specify a TRACE CT,WTRSTART or TRACE CT,WTRSTOP command to start or stop the writer.

CONT or END CONT specifies that the reply continues on another line. Specify END to complete the response.

Displaying component trace status

To display information about the status of the component trace, issue the following command:

```
DISPLAY TRACE,COMP=component_name,SUB=(procedure_jobname)
```

See page 53 for *component_name*.

This command displays information about the status of the component trace for one procedure. To display information about the status of the component trace for all active procedures, enter the following command:

```
DISPLAY TRACE,COMP=component_name,SUBLEVEL,N=8
```

Stopping a component trace

You can stop current tracing with the TRACE CT command:

```
TRACE CT,OFF,COMP=component_name,SUB=(procedure_jobname)
```

See page 53 for *component_name*.

TCP/IP always maintains exception tracing to aid in first failure data capture, even if the trace is turned off.

Obtaining component trace data with a dump

You can request a dump to obtain component trace data for OMPROUTE or a TCP/IP stack.

TCP/IP stack

If an abend occurs in the TCP/IP address space or in a user's address space, TCP/IP recovery dumps the home ASID, primary ASID, secondary ASID, and the TCPIPDS1 dataspace. TCPIPDS1 is the name of the dataspace for each TCP/IP in an MVS image. It contains the trace data for the SYSTCPIP, SYSTCPDA, and SYSTCPIS components.

To view the trace records for a problem where no abend has occurred, use the DUMP command. The following example illustrates an DUMP command:

```
DUMP COMM=(your dump title here)
R n,JOBNAME=tcpiiprocname,DSPNAME='tcpiiprocname'.TCPIPDS1,CONT
R n,SDATA=(nuc,CSA,LSQA,PSA,RGN,SQA,TRT),END
```

To generate a meaningful dump, specify at least *CSA*, *LSQA*, *RGN*, and *SQA*.

OMPROUTE

To obtain a dump of the OMPROUTE address space (which contains the trace table), use the DUMP command, as shown in the following example:

```
DUMP COMM=(enter your dump title here)
R n,JOBNAME=,SDATA(CSA,RGN,ALLPSA,SQA,SUM,TRT,ALLNUC),END
```

RESOLVER

To obtain a dump of the RESOLVER, use the DUMP command, as shown in the following example:

```
DUMP COMM=(enter your dump title here)
R n,JOBNAME=resolver_started_task_name,SDATA=(CSA,RGN,ALLPSA,SQA,SUM,TRT,ALLNUC),END
```

Obtaining component trace data with an external writer

To use an external writer to obtain component trace data for TCP/IP stacks, packet trace, and OMPROUTE, follow the steps below.

1. Enter the appropriate writer procedure in SYS1.PROCLIB, as shown in the following example. Use a separate external writer for each CTRACE component. You can have multiple procedures writing to as many as 16 TRCOUT files either on disk or tape.

```
//PTTCP PROC
//* REFER: SYS1.PROCLIB(PTTCP)
//* COMPID: OPER
```

```

/* DOC:   THIS PROCEDURE IS THE IPCS CTRACE1 EXTERNAL WRITER PROCEDURE.
/*        USED BY TCP/IP   .
/*
/*IEFPROC EXEC PGM=ITTTTCWR,REGION=0K,TIME=1440
/* TIME=1440 to prevent S322 abends
//TRCOUT01 DD DSNAME=MEGA.IPCS.CTRACE1,UNIT=SYSDA,
//          VOL=SER=STORGE,
//          SPACE=(4096,(100,10),,CONTIG),DISP=(NEW,CATLG),
// DCB=(DSORG=PS)
//

```

Note: Do not specify DCB parameters. The external writer defaults to an optimal blocking factor.

2. Start the external writer using the following command:

```
TRACE CT,WTRSTART=procedure_name,WRAP
```

3. Turn the trace on and connect the external writer to the component either by specifying the external writer name in the PARMLIB member, or by specifying the external writer name in the TRACE command. (Since the SYSTCPDA component has no PARMLIB member, the PARMLIB option is not applicable for SYSTCPDA.) To turn the trace on and connect the external writer to the component using a PARMLIB member, add the following TRACEOPTS option to the PARMLIB member:

```
WTR(xxx)
```

where *xxx* is the procedure name of the external writer. Then use this PARMLIB member when starting the program (TCP/IP, OMPROUTE, or the Resolver) or if the program is already executing, issue the command

```
TRACE CT,ON,COMP=component_name,SUB=(procedure_name),PARM=parmlib_member
```

To turn the trace on and connect the external writer without using the PARMLIB member, enter the following command:

```
TRACE CT,ON,COMP=component_name,SUB=(procedure_name)
```

When the system responds, enter the following command:

```
R n,WTR=procedure_name,END
```

where *n* is the response number issued by the system. Note that you can add options to the response. The options vary for each component name. See “Formatting component traces” on page 58 for references to the component options.

4. Use the DISPLAY command to check the external writer status. Include a sublevel.

```

D TRACE,COMP=SYSTCPDA,SUB=(TCPCS2)
IEE843I 11.33.06 TRACE DISPLAY 099
      SYSTEM STATUS INFORMATION
ST=(ON,0064K,00064K) AS=ON BR=OFF EX=ON MT=(ON,064K)
TRACENAME
=====
SYSTCPDA
                                MODE BUFFER HEAD SUBS
                                =====
                                OFF          HEAD    2
      NO HEAD OPTIONS
SUBTRACE                      MODE BUFFER HEAD SUBS
-----
TCPCS2                        ON   0016M

```

```

ASIDS      *NONE*
JOBNAMES   *NONE*
OPTIONS    MINIMUM
WRITER     PTTCP

```

For the TCP/IP CTRACE components, do not be misled by the line in the middle of the display showing the MODE is OFF. This part of the display always says the MODE is OFF because TCP/IP uses the subtrace for all tracing. The subtrace for TCP/CS2 indicates the actual state of the trace. In this example, the trace is active (MODE is ON) with an internal buffer size of 16 M, tracing all ASIDs and all JOBNAMES, using MINIMUM options, and using the external writer PTTCP. Another version of the DISPLAY TRACE command `D TRACE,COMP=component_name,SUBLEVEL,N=8` shows all subtraces for the component.

Note: At this point, the external writer is active for packet, data, and X.25 packet traces.

5. Turn the trace off or disconnect the external writer. The following two commands disconnect from the external writer, while leaving the trace running internally.

```
TRACE CT,ON,COMP=component_name,SUB=(procedure_jobname)
```

When the system responds, enter the second command:

```
R nn,WTR=DISCONNECT,END
```

6. Stop the external writer using the following command:

```
TRACE CT,WTRSTOP=procedure_name
```

Tips for using component trace external writer

- Do not use the same writer to trace more than one TCP/IP stack or OMPROUTE application. If you need to trace multiple stacks or applications, use separate writers.
- If your external writer fills up and the wrap option is on, the writer will overwrite itself. If the nowrap option is on, the writer will stop.
- Use REGION=0K on the trace writer procedure EXEC statement. This will help to ensure that there is enough virtual memory for trace buffers.
- Use TIME=1440 on the EXEC statement. This will prevent S322 abends.
- Use CONTIG on the disk space allocation of the trace data when using the WRAP option. For example: SPACE=(1024,(4096,100),,CONTIG). This will ensure that the space for the trace data set is available.
- Do not specify DCB parameters for trace data sets. The writer will optimize the logical record length and block size for new trace data sets.

Formatting component traces

You can format component trace records using IPCS panels or a combination of IPCS panels and the CTRACE command, either from a dump or from external-writer files. The code for the component trace record formatter can be found in the SYS1.MIGLIB data set. This data set should be added as a concatenation to the STEPLIB data set. For details, refer to the *z/OS MVS IPCS Commands* and the *z/OS MVS IPCS User's Guide*.

IPCS panels

To format component traces using only IPCS panels, follow these steps:

1. Log on to TSO.
2. Access IPCS.

3. Select option 2 (ANALYSIS) from the option list.
4. Select option 7 (TRACES) from the option list.
5. Select option 1 (CTRACE) from the option list.
6. Select option D (Display) from the option list.

The CTRACE DISPLAY PARAMETERS screen is displayed (Figure 4):

```
ITTPC503-----CTTRACE DISPLAY PARAMETERS-----

System      =====>          (System name or blank)
Component   =====>          (Component name (required))
Subnames    =====>

GMT/LOCAL   =====>          (Greenwich Mean Time or Local; GMT is default)
Start time  =====>          (mm/dd/yy, hh:mm:ss.dddddd)
Stop time   =====>
Limit       =====>          Exception =====>
Report type =====> FULL      (Short, Summary, Full, Tally)
User exit   =====>          (Exit program name)
Override source =====>
Options     =====>

To enter/verify required values, type any character
Entry IDS =====>      Jobnames =====>      ASIDs =====>      OPTIONS =====>      SUBS =====>

CTRACE COMP(xx) FULL

COMMAND =====>
      F1=Help  F2=Split  F3=End   F4=RETURN  F5=RFIND   F6=MORE   F7=UP
      F8=DOWN  F9=Swap   F10=LEFT F11=RIGHT F12=CURSOR
```

Figure 4. IPCS CTRACE

Enter the component name in the COMPONENT field and as the value in COMP(xx). For descriptions of options, see the following sections:

- SYSTCPDA: See Chapter 5, “TCP/IP services traces and IPCS support” on page 53.
- SYSTCPIP: See Chapter 5, “TCP/IP services traces and IPCS support” on page 53.
- SYSTCPIS: See Chapter 5, “TCP/IP services traces and IPCS support” on page 53.
- SYSTCPRE: See Chapter 32, “Diagnosing resolver problems” on page 683.
- SYSTCPRT: See “TCP/IP services component trace for OMPROUTE” on page 615.

CTTRACE command

To format component traces using the CTRACE command, follow these steps:

1. Log on to TSO.
2. Access IPCS.
3. Select option 6 (COMMAND) from the option list.
4. Enter a CTRACE command and options on the IPCS command line.

Syntax

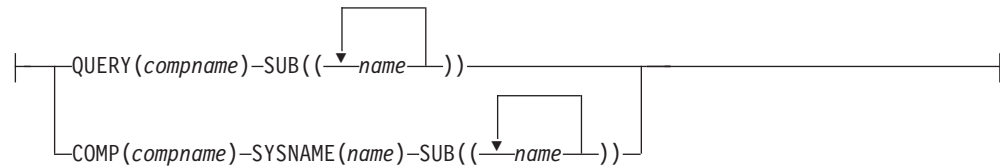
Following is the syntax of the IPCS CTRACE command:

CTTRACE syntax

```
▶▶CTTRACE—| Component selection |—| Report type |—————▶
```

► Data selection | Address space selection | Setdef parameters | ►

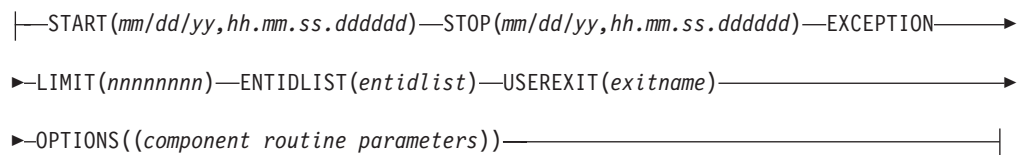
Component Selection:



Report Type:



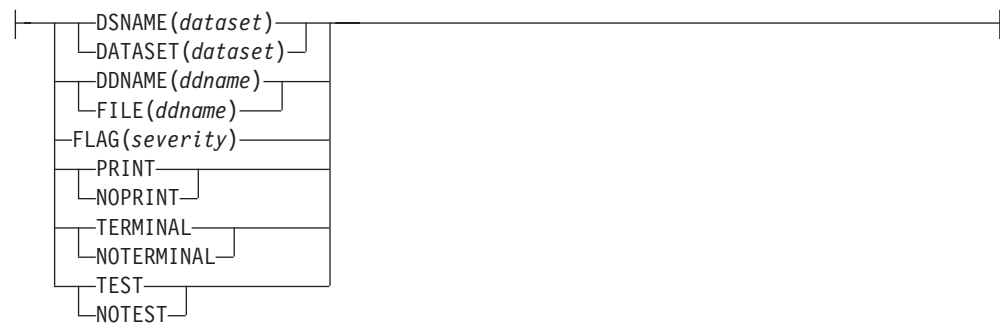
Data Selection:



Address Space Selection:



Setdef Parameters:



Parameters

Refer to *z/OS MVS IPCS Commands* for details on the CTRACE parameters.

Keywords

The following CTRACE keywords can be used with TCP/IP component trace formatters

JOBLIST, JOBNAME

The JOBLIST and JOBNAME keywords can be used to select packet trace records with a matching link name. However, only the first eight characters of the link name are matched and no asterisks are accepted in the job name. They can also be used to match the job name in data trace records.

ASIDLIST

The ASIDLIST can be used to select trace records only for a particular address space.

GMT

The time stamps are converted to GMT time.

LOCAL

The time stamps are converted to LOCAL time.

SHORT

If the OPTIONS string does not specify any reports, then format the trace records. Equivalent to the FORMAT option.

FULL

If the OPTIONS string does not specify any reports, then format and dump the trace records. Equivalent to the FORMAT and DUMP options.

SUMMARY

If the OPTIONS string does not specify any reports, then create a one line summary for each trace record. Equivalent to the SUMMARY option.

TALLY

If the OPTIONS string does not specify any reports, then count the trace records.

START and STOP

These keywords limit the trace records that are seen by the formatter. The STOP keyword determines the time when records are no longer seen by the packet trace report formatter.

Note: CTRACE always uses the time the trace record was moved to the buffer for START and STOP times.

LIMIT

Determines the number of records the formatter is allowed to process.

USEREXIT

The CTRACE USEREXIT is called for TCP/IP formatters, except packet trace. Therefore, the packet trace formatter calls the CTRACE USEREXIT before testing the records with the filtering criteria. If it returns a nonzero return code, then the record will be skipped. The USEREXIT can also be used in the OPTIONS string. It is called after the record has met all the filtering criteria in the OPTIONS string. For details, see "Formatting packet traces using IPCS" on page 95.

Examples

The following example shows the error message when the specified address space is not available in the dump.

```
CTRACE QUERY(SYSTCPIP) SUB((TCPSVT1)) FULL LOCAL  
COMPONENT TRACE QUERY SUMMARY
```

```
ITT10003I There are no trace buffers in the dump for COMP(SYSTCPIP)SUB((TCPSVT1))
```

The following example shows the results when the CTRACE QUERY command is issued for a dump when the address space is available. Note that the first options is the relevant one (ignore the second options list). The buffer size and options list are displayed only for a dump data set, not an external writer data set.

```
CTRACE QUERY(SYSTCPIP) SUB((TCPSVT2)) FULL LOCAL  
COMPONENT TRACE QUERY SUMMARY
```

```
COMP(SYSTCPIP)SUBNAME((TCPSVT2))
```

```
START = MT 02/21/2001 15:25:49.432 LOCAL  
STOP = 180 02/21/2001 15:51:16.8  
Buffer size: 0050M
```

```
OPTIONS: CONFIG,CCKET,FIREWALL,IOCTL,MESSAGE,OETCP,OPCMDS,  
OPMSGs,PASAPI,PING,SOCKAPI,TN,UDP,XCF,CLAW,INT  
ERNET,LCS,VTAM,VTAMDATA
```

```
OPTIONS: MINIMUM
```

Event trace (SYSTCPIP) for TCP/IP stacks

Event trace for TCP/IP stacks traces individual TCP/IP components (such as STORAGE, INTERNET, and so forth) and writes the information either to a data set (using an external writer), or internally to the TCP/IP dataspace (TCPIPDS1). To aid in first failure data capture, a minimal Component Trace is always started during TCP/IP initialization if you use the TCP/IP Component Trace SYS1.PARMLIB member, CTIEZBxx.

You can select trace records at run time by any of the following methods:

- JOBNAME
- Address space identifiers (ASID)
- Trace option
- IP address
- Port number

If using the TELNET options, do not specify the JOBNAME parm when starting CTRACE.

Specifying trace options

You can specify component trace options at TCP/IP initialization or after TCP/IP has initialized.

Specifying trace options at initialization

To start TCP/IP with a specific trace member, use the following command:

```
S tcip_procedure_name,PARMS=CTRACE(CTIEZBxx)
```

where CTIEZBxx is the component trace SYS1.PARMLIB member. You can create this member yourself or you can update the default SYS1.PARMLIB member, CTIEZB00 (see Figure 5 on page 63). For a description of trace options available in the CTIEZB00, see Table 11 on page 66.

Note: Besides specifying the desired TCP/IP traces, you can also change the component trace buffer size. The buffer size can be changed only at TCP/IP initialization.

```

/*****
/*
/* Communications Server for OS/390
/* SMP/E Distribution Name: CTIEZB00
/*
/* MEMBER: CTIEZB00
/*
/*
/* Copyright: Licensed Materials - Property of IBM
/*
/* 5694-A01
/*
/* (C) Copyright IBM Corp. 1996, 2002
/*
/*
/* STATUS = CSV1R4
/*
/* DESCRIPTION = This parmlib member causes component trace for
/* the TCP/IP product to be initialized with a
/* trace buffer size of 8 megabytes.
/*
/* This parmlib members only lists those TRACEOPTS
/* value specific to TCP/IP. For a complete list
/* of TRACEOPTS keywords and their values see
/* OS/390 MVS INITIALIZATION AND TUNING REFERENCE.
/*
/* $MAC(CTIEZB00) PROD(TCPIP): Component Trace SYS1.PARMLIB member
/*
*****/

```

Figure 5. SYS1.PARMLIB member CTIEZB00 (Part 1 of 3)

```

TRACEOPTS
/* ----- */
/*   ON OR OFF: PICK 1                               */
/* ----- */
/*           ON                                       */
/*           OFF                                       */
/* ----- */
/*   BUFSIZE: A VALUE IN RANGE 128K TO 256M          */
/* ----- */
/*           BUFSIZE(8M)                             */
/*           JOBNAME(jobname1,...)                   */
/*           ASID(Asid1,...)                         */
/*           WTR(wtr_procedure)                      */
/* ----- */
/*   Note, the following groups of trace options are supported: */
/* ----- */
/*   ALL      = All options except ROUTE, SERIAL, SOCKAPI, STORAGE */
/*             and TIMER                                           */
/* ----- */
/*   CSOCKET  = PFS + SOCKET                                       */
/* ----- */
/*   DLC      = CLAW + INTERNET + LCS + VTAM + VTAMDATA           */
/* ----- */
/*   IN       = CONFIG + INIT + IOCTL + OPCMDS + OPMSGs           */
/* ----- */
/*   LATCH    = SERIAL                                             */
/* ----- */
/*   MINIMUM  = INIT + OPCMDS + OPMSGs                             */
/* ----- */
/*   OETCP    = ENGINE + PFS + QUEUE + TCP                        */
/* ----- */
/*   OEUDP    = ENGINE + PFS + QUEUE + UDP                        */
/* ----- */
/*   PING     = ARP + ICMP + RAW + ND                             */
/* ----- */
/*   RW       = ENGINE + PFS + QUEUE + RAW + SOCKET               */
/* ----- */
/*   SMTP     = ENGINE + IOCTL + PASAPI + PFS + QUEUE + SOCKET + TCP */
/* ----- */
/*   SYSTEM   = INIT + OPCMDS + OPMSGs + SERIAL + STORAGE + TIMER + */
/*             WORKUNIT                                             */
/* ----- */
/*   TC       = ENGINE + PFS + QUEUE + SOCKET + TCP               */
/* ----- */
/*   TN       = PFS + TCP + TELNET + TELNVTAM                     */
/* ----- */
/*   UD       = ENGINE + PFS + QUEUE + SOCKET + UDP               */
/* ----- */
/* ----- */
/*   OPTIONS: NAMES OF FUNCTIONS OR GROUPS TO BE TRACED:         */
/* ----- */

```

Figure 5. SYS1.PARMLIB member CTIEZB00 (Part 2 of 3)

```

/* ----- */
/*      OPTIONS(      */
/*          'ALL      */
/*      , 'ACCESS    */
/*      , 'AFP       */
/*      , 'ARP       */
/*      , 'CLAW      */
/*      , 'CONFIG    */
/*      , 'CSOCKET   */
/*      , 'DLC       */
/*      , 'ENGINE    */
/*      , 'FIREWALL  */
/*      , 'ICMP      */
/*      , 'IN        */
/*      , 'INIT      */
/*      , 'INTERNET  */
/*      , 'IOCTL     */
/*      , 'IPADDR(nnn.nnn.nnn.nnn/mmm.mmm.mmm, */
/*              nnn.nnn.nnn.nnn/pp,          */
/*              hhhh:hhhh/ppp)              */
/*      , 'LATCH     */
/*      , 'LCS       */
/*      , 'MESSAGE   */
/*      , 'MINIMUM   */
/*      , 'ND        */
/*      , 'NONE      */
/*      , 'OETCP     */
/*      , 'OEUDP     */
/*      , 'OPCMDS    */
/*      , 'OPMSGs    */
/*      , 'PASAPI    */
/*      , 'PFS       */
/*      , 'PING      */
/*      , 'POLICY    */
/*      , 'PORT(pppp,oooo,rrrrr,tttt) ' */
/*      , 'QUEUE     */
/*      , 'RAW       */
/*      , 'ROUTE     */
/*      , 'RW        */
/*      , 'SERIAL    */
/*      , 'SMTP      */
/*      , 'SNMP      */
/*      , 'SOCKAPI   */
/*      , 'SOCKET    */
/*      , 'STORAGE   */
/*      , 'SYSTEM    */
/*      , 'TC        */
/*      , 'TCP       */
/*      , 'TELNET    */
/*      , 'TELNETAM  */
/*      , 'TIMER     */
/*      , 'TN        */
/*      , 'UD        */
/*      , 'UDP       */
/*      , 'VTAM      */
/*      , 'VTAMDATA  */
/*      , 'WORKUNIT  */
/*      , 'XCF       */
/*      )           */

```

Figure 5. SYS1.PARMLIB member CTIEZB00 (Part 3 of 3)

Table 11 on page 66 describes the available trace options and groups. A group turns on multiple trace options. The group name identifies traces that should be on for a specific problem area. Trace groups provide a way to collect trace data by problem

type.

Table 11. Trace options

Trace Event	Description
ALL	All types of records except ROUTE, SERIAL, STORAGE, and TIMER. Slow Performance: Using this option will slow performance considerably, so use with caution.
ACCESS	Trace creation, modification, and manipulation of the Network Access tree, along with results of all Network Access queries.
AFP™	Turns on trace for fast response cache accelerator.
ARP	Shows address resolution protocol (ARP) cache management and ARP timer management. This option also shows all outbound and inbound ARP packets. Note: The information provided differs depending on the type of device. Refer to the <i>z/OS Communications Server: IP Configuration Guide</i> for information on what is supported for each device. Note: The ARP and ND options are aliases. If you turn one on, you turn on the other option, and if you turn one off, you turn off the other option. When formatting the trace, these options can be filtered separately.
CLAW	Shows all control flows for a CLAW device.
CONFIG	Turns on trace for configuration updates.
CSOCKET	Turns on the following trace options: <ul style="list-style-type: none">• PFS• SOCKET
DLC	Turns on the following trace options: <ul style="list-style-type: none">• CLAW• INTERNET• LCS• VTAM• VTAMDATA
ENGINE	Turns on trace for stream head management. Note: The ENGINE and QUEUE options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately.
FIREWALL	Turns on trace for firewall events.
ICMP	Turns on trace for the ICMP protocol.
ICMPv6	Turns on trace for the ICMPv6 protocol.
IN	Turns on the following trace options: <ul style="list-style-type: none">• CONFIG• INIT• IOCTL• OPCMDS• OPMSGs

Table 11. Trace options (continued)

Trace Event	Description
INIT	Turns on trace for TCP/IP Initialization/Termination. Note: The INIT, OPCMDS, and OPMGS options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately.
INTERNET	Turns on trace for Internet Protocol layer. Slow Performance: Using this option slows performance considerably, so use with caution.
IOCTL	Turns on trace for IOCTL processing.
IPADDR=(list)	Turns on trace by IP address.
LATCH	Turns on the following trace option: • SERIAL
LCS	Shows all control flows for an LCS device.
MESSAGE	Turns on trace for message triple management. Slow Performance: Using this option slows performance considerably, so use with caution.
MINIMUM	Turns on the following trace options: • INIT • OPCMDS • OPMGS
NONE	Turn off all traces but exception traces, which always stay on.
ND	Enable Neighbor Discovery trace option. Note: The ARP and ND options are aliases. If you turn one on, you turn on the other option, and if you turn one off, you turn off the other option. When formatting the trace, these options can be filtered separately.
OETCP	Turns on the following trace options: • ENGINE • PFS • QUEUE • TCP
OEUDP	Turns on the following trace options: • ENGINE • PFS • QUEUE • UDP
OPCMDS	Turns on traces of operator commands. Note: The INIT, OPCMDS, and OPMGS options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately.

Table 11. Trace options (continued)

Trace Event	Description
OPMSGS	Turns on message trace for console messages. Note: The INIT, OPCMDS, and OPMGS options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately.
PASAPI	Turns on traces for transforms that handle Pascal APIs.
PFS	Turns on trace for the physical file system layer.
PING	Turns on the following trace options: <ul style="list-style-type: none"> • ARP • ICMP • RAW
POLICY	Trace the stack usage of Policy Rules and Actions.
PORT=(list)	Turns on trace by port number.
QUEUE	Turns on trace for stream queue management. Note: The ENGINE and QUEUE options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately.
RAW	Turns on trace for the RAW transport protocol.
ROUTE	Trace manipulation of IP Routing Tree.
RW	Turns on the following trace options: <ul style="list-style-type: none"> • ENGINE • PFS • QUEUE • RAW • SOCKET
SERIAL	Turns on trace for lock obtain and release. Slow Performance: Using this option slows performance considerably, so use with caution.
SMTP	Turns on the following trace options: <ul style="list-style-type: none"> • ENGINE • IOCTL • PASAPI • PFS • QUEUE • SOCKET • TCP
SNMP	Turns on trace for SNMP SET requests.
SOCKAPI	Trace Macro and Call Instruction API calls (see “Socket API traces” on page 74)
SOCKET	Turns on trace for the Sockets API layer.
STORAGE	Turns on trace for storage obtain and release. Slow Performance: Using this option slows performance considerably, so use with caution.

Table 11. Trace options (continued)

Trace Event	Description
SYSTEM	Turns on the following trace options: <ul style="list-style-type: none"> • INIT • OPCMDS • OPMSGGS • SERIAL • STORAGE • TIMER • WORKUNIT
TC	Turns on the following trace options: <ul style="list-style-type: none"> • ENGINE • PFS • QUEUE • SOCKET • TCP
TCP	Turns on trace for the TCP transport protocol. Slow Performance: Using this option slows performance considerably, so use with caution.
TELNET	Turns on trace for TELNET events.
TELNVTAM (an alias for TELNET)	Turns on trace for TELNET events.
TIMER	Turns on trace for TCP timers. Slow Performance: Using this option slows performance considerably, so use with caution.
TN	Turns on the following trace options: <ul style="list-style-type: none"> • PFS • TCP • TELNET • TELNVTAM
UD	Turns on the following trace options: <ul style="list-style-type: none"> • ENGINE • PFS • QUEUE • SOCKET • UDP
UDP	Turns on trace for UDP transport protocol. Slow Performance: Using this option slows performance considerably, so use with caution.
VTAM	Shows all of the nondata-path signaling occurring between IF and VTAM.
VTAMDATA	Shows data-path signaling between IF and VTAM, including a snapshot of media headers and some data. Slow Performance: Using this option slows performance considerably, so use with caution.
WORKUNIT	Turns on trace for work unit scheduling.
XCF	Turns on trace for XCF events.

Specifying trace options after initialization

After TCP/IP initialization, you must use the TRACE CT command to change the component trace options. All options except for the size of the TCP/IP component trace buffer can be changed using this command. Each time a new component trace is initiated, all prior trace options are turned OFF and the new traces are activated.

You can specify TRACE CT with or without the PARMLIB member. For details, see Chapter 5, “TCP/IP services traces and IPCS support” on page 53.

Filter TCPIP CTRACE by IP address

To execute a trace on a particular IP address, use the IP address, port number, asid, and JOBNAME as targets for filtering the records.

To use this function, start by issuing the TRACE command:

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpip_procedure_name)

R 01, OPTIONS = (IPADDR=(12AB:0:0:CD30::/60),PORT=(1012))
R 02,OPTIONS=(ENGINE,PFS),END
```

Trace records of type ENGINE or PFS for an IP address of 192.48.32.37 and a port number of 80 or 33 will be captured. The IP address used is the foreign session partner IP address. The local port number is the local session partner port number. The choice of the IP and Port numbers is determined by the direction of the data. When filters are used the trace record must be accepted by each filter. Each filter can specify multiple values (up to 16). The trace record must match one of the values.

Inbound

Data received at the IP layer is considered inbound data. The source IP address and the destination port number are used.

Outbound

Data sent in the PFS layer is considered outbound data. The destination IP address and the source port number are used.

There are five criteria for selecting trace records for recording: TYPE, JOBNAME, ASID, IPADDR, and PORT. Each criterion can specify one or more values. If a criterion has been specified, the record to be traced must match one of the values for that criterion. If a criterion has not been specified, the record is not checked and does not prevent the record from being recorded. However, the record must match all specified criteria. In the above example, JOBNAME and ASID were not specified, so the value of JOBNAME and ASID in the record are not checked.

Note: There is an exception for IPADDR and PORT. Some trace records do have a IP address or a port number. Therefore, the IP address is only checked if it is nonzero and the port number is checked only if it is nonzero.

You can also specify a range of IP addresses to trace. For example,

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(TCPIP_PROC_NAME)
R xx, OPTIONS = (IPADDR={nnn.nnn.nnn.nnn{,nnn.nnn.nnn.nnn/mmm.mmm.mmm.mmm,nnn.nnn.nnn.nnn/dd,hhhh::hhhh/ddd}},
PORT = {pppp{,pppp}})
```

IPADDR

An IP address. Up to 16 addresses can be specified. IPv4 addresses are in dotted decimal notation, for example: 192.48.24.57. IPv6 addresses are in colon hexadecimal notation or in a combination of both colon hexadecimal and dotted decimal for IPv4-mapped IPv6 addresses, for example:

beef::c030:1839. An IP address of 0 can be used for trace records that do not have an IP address. A subnet mask is indicated by a slash (/) followed by the prefix length in decimal or by a dotted decimal subnet mask for IPv4 addresses. The prefix length is the number of one bits in the mask. For IPv4 addresses it might be in the range of 1–32; for IPv6 addresses it might be in the range of 1–128, for example: 192.48.24/24 or FEC0::0/10

PORT The list of port numbers to be filtered. Up to 16 port numbers can be specified. The port numbers, specified in decimal, must be in the range 0–65535. A trace record with a zero port number is not subject to port number filtering.

Notes:

1. The equal sign (=) after the IPADDR and PORT keywords is optional.
2. You can specify the IPADDR and PORT keywords multiple times in an OPTIONS string. If you do, all the values will be saved.
3. All the values in the OPTIONS keyword must be specified in one trace command. The next trace command with an OPTIONS keyword replaces all the options specified.

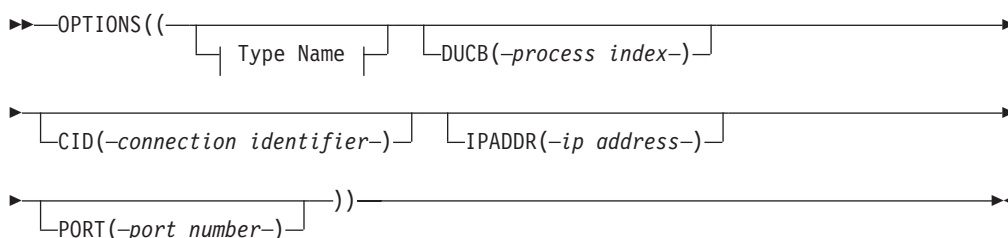
Formatting event trace records for TCP/IP stacks

You can format event trace records using IPCS panels or a combination of IPCS panels and the CTRACE command. For a description of the relevant IPCS panels, see Chapter 5, “TCP/IP services traces and IPCS support” on page 53. For more information about other CTRACE options, refer to the *z/OS MVS IPCS Commands*.

When using an IPCS panel, enter the trace types in the following format:

option DUCB() CID()

Following is the syntax for the CTRACE command for TCP/IP stacks. For more information on the command and IPCS, refer to the *z/OS MVS IPCS User's Guide*.



Type Name

The name of a trace type. Only records of these types are formatted. For a list of types, see Table 11 on page 66.

DUCB A process index for the thread of execution. Up to 16 indexes can be specified. The DUCB index values can be entered either in decimal (such as DUCB(18)) or hexadecimal (such as DUCB(X'12')), but are displayed in hexadecimal format.

CID	A connection identifier. Up to 16 identifiers can be specified. The CID values can be entered in either decimal (such as CID(182)) or hexadecimal (such as CID(X'0006CE7E')), but are displayed in hexadecimal. This is the same value that appears in the NETSTAT connections display.
------------	---

IPADDR

An IP address. Up to 16 addresses can be specified. IPv4 addresses are in

dotted decimal notation, for Example: 192.48.24.57. IPv6 addresses are in colon hexadecimal notation or in a combination of both colon hexadecimal and dotted decimal for IPv4-mapped IPv6 addresses, for example: beef::c030:1839. An IP address of 0 can be used for trace records that do not have an IP address. A subnet mask is indicated by a slash (/) followed by the prefix length in decimal or by a dotted decimal subnet mask for IPv4 addresses. The prefix length is the number of one bits in the mask. For IPv4 addresses it might be in the range of 1–32; for IPv6 addresses it might be in the range of 1–128, for example: 192.48.24/24 or FEC0::0/10

PORT A port number. Up to 16 port numbers can be specified. Note that the port numbers can be entered in decimal, such as PORT(53), or hexadecimal, such as PORT(x'35'), but are displayed in decimal. These are port numbers in the range 0–65535. A port number of 0 can be used for trace records that do not have a port number.

Note: Standard TSO syntax is used for the keywords and their values. For example, CID(1 2 3).

Figure 6 on page 73 contains the start of the CTRACE formatted output. The CTRACE command parameters are followed by the trace date and column headings. Then there is one TCP/IP CTRACE record with four data areas. The parts of the TCP/IP CTRACE record are:

- 1** Standard IPCS header line, which includes the system name (VIC142), TCP/IP option name (PFS), time stamp, and record description.
- 2** TCP/IP header line with address space and user (or job name) information.
- 3** TCP/IP header line with task and module information.
- 4** TCP/IP header line with session information (CID, IP address, and port number).
- 5** TCP/IP header line for a data area. This line has the address (first four bytes are the ALET), the length of data traced, and the data description. Following the description, the actual data is in dump format (hexadecimal offset, hexadecimal data, and EBCDIC data).
- 6** There are four data areas in this example. The third one ("Return Value Errno ErrnoJr") has an extra line. The ERRNO line is added only when the return value is -1 and the ERRNO indicates an error. In this example, the return code is hexadecimal 462 (decimal 1122). Refer to the *z/OS Communications Server: IP and SNA Codes* for more information.
- 7** TCP/IP trailer and separator line with the record sequence number (hexadecimal 573E).

COMPONENT TRACE FULL FORMAT
COMP(SYSTCPIP)SUBNAME((TCPSVT))

**** 11/03/1999

	SYSNAME	MNEMONIC	ENTRY ID	TIME STAMP	DESCRIPTION
1	VIC142	PFS	60010018	14:57:59.207826	Socket IOCTL Exit
2	HASID..001E	PASID..000E	SASID..001E	USER...OMPROUTE	
3	TCB...007E7A68	MODID..EZBPFI0C	REG14..161D86C0	DUCB...0000000C	
4	CID...0000003A	PORT.....0			
	IPADDR. 3F98::D002:A521				
5	ADDR...00000000	14D9EED0	LEN....000000A0	OSI	
6	+0000	D6E2C940	000000A0	00000000	00000000 OSI
	+0010	0500001B	14D9EF70	00500AC8	00000000 R...&.H....
	+0020	00000000	00000000	00000000	00000000
	+0030	00000000	00000000	00000000	00281080
	+0040	14D9FC0C	00000C00	14D9FFE8	00000000 .R.....R.Y....
	+0050	00000000	00000000	00000000	00000000
	+0060	00000000	00000000	00000000	00000000
	+0070	00000000	00000000	00000000	00000000
	+0080	00000000	00000000	00000000	00000000
	+0090	00000000	00000000	00000000	00000000
	ADDR...00000000 12D7F874 LEN....00000004 SCB Flags				
	+0000	00280000		
	ADDR...00000000 12E88598 LEN....00000010 Return Value Errno ErrnoJr				
	+0000	C5D9D9D5	FFFFFFFF	00000462	740E006B ERRN.....,
	ERRNO..-1, 462, 740E006B				
	ADDR...00000000 14D9F4E4 LEN....00000048 IOCTL Request				
	+0000	C3C6C7D4	D9C5D840	0000008E	00000462 CFGMREQ
	+0010	00000320	00000500	00000000	00000000
	+0020	740E0005	00000000	14B4C7C0	00000000 G{....
	+0030	00000000	00050063	00000000	00000000
	+0040	F3F1F0F1	00000000		3101....
7	=====0000573E				

Figure 6. Start of component trace full format

Additional fields in CTRACE output

The ERRNO line in Figure 6 is one of two cases in which the formatter extracts data and formats it in a special way. The other case is for "TCB CTRL" and "IUDR" data. Several fields are copied from the data and formatted with character interpretation of fields, such as converting values to decimal or dotted decimal. Figure 7 on page 74 is an example. Note the additional fields (TcpState, TpiState, and others) following the hexadecimal data.

```

BOTSWANA TCP          40030002 20:51:35.652462 Select/Poll Exit Detail
HASID..0082          PASID..0088          SASID..000E          USER...POLAGENT
TCB...007E4640 MODID..EZBTCFSP REG14..10FD7C5E DUCB...00000016
CID...000004DC PORT....1925
IPADDR. 197.011.106.001
ADDR...00000000 116B04DC LEN....00000004 Select function code
+0000 00000002 | .... |
ADDR...00000000 116B0668 LEN....00000004 Output condition indicators
+0000 40000000 | ... |
ADDR...00000000 7F60C508 LEN....000003D8 Transmission Control Block
+0000 E3C3C240 C3E3D9D3 00050009 81801000 | TCB CTRL....a... |
+0010 00000000 00000000 00000000 138C4F08 | .....|. |
...
+0170 00000000 00020000 00003000 45000028 | ..... |
+0180 1CB14000 40060000 C50B6A01 C50B6A01 | .. . ...E...E... |
+0190 00000000 00000000 00000000 00000000 | ..... |
+01A0 00000000 00000000 00000000 00000000 | ..... |
+01B0 00000000 00000000 0000FFFF FFFF4000 | ..... |
+01C0 00000000 00000000 00000000 00000001 | ..... |
+01D0 07850185 F4258CA0 F425A310 50107F32 | .e.e4...4.t.&." |
+01E0 00000000 0004FFCB 01030300 0101080A | ..... |
...
+03D0 010E1301 0E21010E | ..... |
TcpState..ESTAB TpiState..WLOXFER
SrcPort..1925 SrcIPAddr. 197.11.106.1
DstPort..389 DstIPAddr. 197.11.106.1
FLAGS.....ACK

```

Figure 7. Component trace full format showing character interpretation of fields

Socket API traces

The SOCKAPI option, for the TCP/IP CTRACE component SYSTCPIP, is intended to be used for application programmers to debug problems in their applications. The SOCKAPI option captures trace information related to the socket API calls that an application may issue. The SOCKET option is primarily intended for use by TCP/IP Service and provides information meant to be used to debug problems in the TCP/IP socket layer, UNIX System Services, or the TCP/IP stack.

CTRACE is available only to users with console operator access. If the application programmer does not have console access, someone must provide the CTRACE data to the programmer. For security reasons, it is suggested that only the trace data related to the particular application be provided. The following sections explain how to obtain the trace data for a particular application, format it, and save the formatted output. The application data can be isolated when recording the trace, or when formatting it, or both.

z/OS provides several socket APIs that can be used by applications. Figure 8 on page 75 shows different APIs along with the high level flows of how they interact with the TCP/IP stack. The SOCKAPI trace output is captured in the Sockets Extended Assembler Macro API (the "Macro API"). Given the structure of the TCP/IP APIs, this trace also covers the Call Instruction API, the CICS socket API, and the IMS socket API. Some of the socket APIs based on the Macro API currently encapsulate some of the Macro API processing. For example, in a CICS environment, CICS transactions do not issue an INITAPI call. Rather, this is done automatically for the socket API by the TCP/IP CICS TRUE (Task Related User

Exit) component layer. If the socket API trace is active, trace records for the INITAPI calls are created.

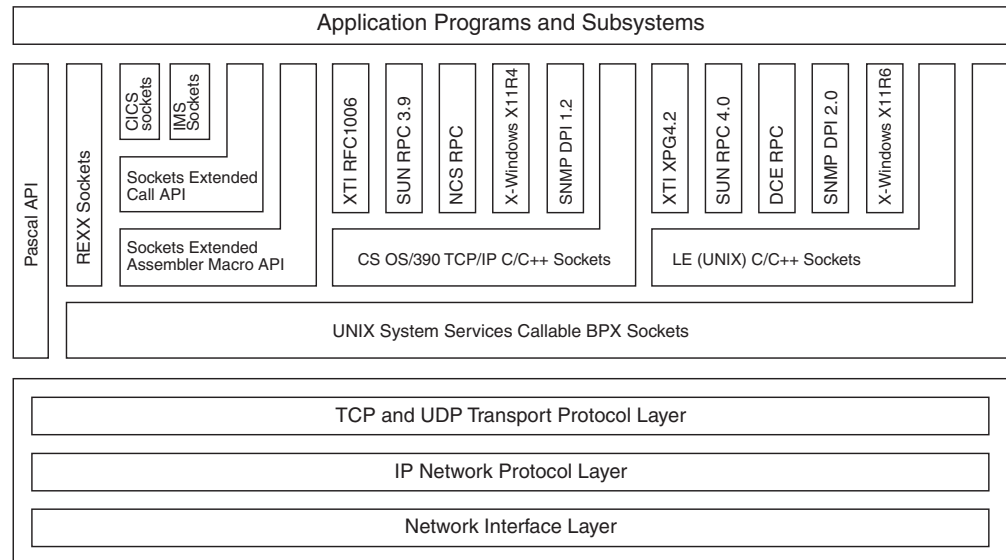


Figure 8. TCP/IP networking API relationship on z/OS

Recommended options for the application trace

The CTRACE facility has flexibility such as filtering, combining multiple concurrent applications and traces, and using an external writer. Some things to keep in mind when using CTRACE:

- Even though the CTRACE can be used to trace multiple applications at the same time and in conjunction with other trace options, it is not recommended. Multiple traces make problem determination more difficult.
- For performance reasons, the data being recorded should be filtered, to minimize the overhead of recording the trace, to make formatting faster, to save storage, and to minimize wrapping (overwriting of older trace records by new trace records).

Optimally, use the CTRACE facility to capture all the SOCKAPI trace records for one application. The trace can be filtered various ways when formatting. If necessary, you can limit the trace data collected by IP address or port number, but you risk some records not being captured. For example, the problem may be that the wrong IP address or port number was coded or used. Both the IP address and port number are formatting options.

Recommended options for optimally capturing the application data follow.

- **Trace only one application.** Use the job name or ASID option when capturing the trace to limit the trace data to one application.
- **Trace only the SOCKAPI option.** To get the maximum number of SOCKAPI trace records, specify only the SOCKAPI option. (You will also get exception records. Exception records are always traced because they are considered unusual events.)
- **Use an external writer.** The external writer is recommended:
 - To separate the SOCKAPI trace records from other internal data that exist in a dump (for security and other reasons)
 - To avoid interrupting processing with a dump of the trace data

- To keep the buffer size from limiting the amount of trace data
- To avoid increasing the buffer size (which requires restarting TCP/IP)
- To handle a large number of trace records
- **Trace only one TCP/IP stack.** If you are running with multiple TCP/IP stacks on a single z/OS image, use the external writer for only one TCP/IP stack.
- **Activate the data trace only if more data is required.** The SOCKAPI trace contains the first 96 bytes of data sent or received, which is usually sufficient. If additional data is needed, the data trace records can be correlated with the SOCKAPI records.

How to collect the SOCKAPI trace option

The existing CTRACE facility for TCP/IP's SYSTCPIP component is used for the SOCKAPI trace option. Collecting the trace is described generally in "Component trace" on page 53. This section describes how to collect the trace for use by application programmers.

The trace can be started automatically when TCP/IP starts or can be started or modified while TCP/IP is executing. A CTRACE PARMLIB member is required for starting the trace automatically, and can optionally be used after TCP/IP has been started.

CTRACE PARMLIB member CTIEZBxx: Sample member CTIEZB00 is shipped with TCP/IP. The most important option is BUFSIZE because it cannot be changed after TCP/IP is started. All the other options can be changed while TCP/IP is active. To get SOCKAPI records, you must specify either the SOCKAPI or ALL option. It is recommended that you use the ALL option only if necessary as the output can be very large.

Note: The BUFSIZE is the size of the internal tracing buffer and is not relevant if you are using the external writer.

TCP/IP start procedure: The CTRACE PARMLIB member can be specified in the TCP/IP start procedure or on the START command. The sample TCPIPROC start procedure specifies member name CTIEZB00. Specifying the member name on the START command depends on how the TCP/IP start procedure is coded. Following is an example of overriding the PARMLIB member name using the sample TCPIPROC start procedure.

```
S TCPIPROC,PARM='CTRACE(CTIEZBAN)'
```

To specify the suffix of the SYS1.PARMLIB member for SYSTCPIP CTRACE initialization, use the TRC option. The TRC option appends the two letters to CTIEZB. The full member name will be "CTIEZBxx". The default value is "00". In this example, the PARMLIB member for SYSTCPIP is CTIEZBAN, an equivalent command is

```
S TCPIPROC,PARM='TRC=AN'
```

To specify the suffix of the SYS1.PARMLIB member for SYSTCPIP CTRACE initialization, use the IDS option. The IDS option appends the two letters to CTIIDS. The full member name is "CTIIDSxx". The default value is "00".

```
S TCPIPROC,PARM='IDS=AN'
```

You can specify multiple parameters. If you specify both the CTRACE and TRC parameters, whichever appears last in the parameter string will be used.

TRACE command: To start, modify, or stop the trace after TCP/IP has been started, use the MVS TRACE command. The TRACE command replaces all prior settings except the buffer size (which is set during TCP/IP initialization and cannot be modified). When modifying the options, be sure to specify the SOCKAPI option. Following are some examples showing how to start the trace. The SUB option is the subtrace name, which for TCP/IP is the job name of the stack (usually this is the TCP/IP start procedure name). In these examples, the subtrace is TCPIPROC, the name of the sample procedure. In the examples, variable fields are in lowercase.

- Example 1. Activating the trace with just the SOCKAPI option:

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpiproc)
R n,JOBNAME=(ezasokjs),OPTIONS=(sockapi),end
```

- Example 2. Specifying a PARMLIB member, which contains the trace options:

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpiproc),PARM=ctiezban
```

To stop the trace, you can either use the TRACE CT,OFF command or you can reissue TRACE CT,ON with different parameters. Following is an example of the OFF option:

```
TRACE CT,OFF,COMP=SYSTCPIP,SUB=(tcpiproc)
```

When using the TRACE command, be sure to notice message ITT038I, which indicates whether the command was successful or not:

```
14.11.29 ITT038I NONE OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT
COMMAND WERE SUCCESSFULLY EXECUTED.
or
14.11.40 ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT
COMMAND WERE SUCCESSFULLY EXECUTED.
```

Refer to *z/OS MVS System Commands* for more information about the TRACE command.

External writer: If the trace is active, it is always written to an internal buffer (whose size is set to BUFSIZE during TCP/IP initialization). The internal buffer is available only in a dump of TCPIP and its dataspace (TCPIPDS1). Optionally, the trace can also be written to an external data set using the MVS CTRACE external writer. If you use an external writer, the trace records are copied to a data set.

To use an external writer, you must create a procedure which specifies the job to run (the external writer) and the trace output data sets. There is an example in Chapter 5, “TCP/IP services traces and IPCS support” on page 53. Also, refer to *z/OS MVS Diagnosis: Tools and Service Aids* for more information about CTRACE, the external writer (including a sample procedure), dispatching priority for the external writer job, and wrapping.

The external writer must be started before the trace can be activated. The trace must be inactivated before the writer can be stopped. The writer must be stopped before the data set can be formatted or transferred. For example, here is a sequence of commands for using an external writer procedure named “ctw”:

```
TRACE CT,WTRSTART=ctw
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpiproc)
R n,JOBNAME=(ezasokjs),OPTIONS=(sockapi),WTR=ctw,end
```

<run application being traced>

```
TRACE CT,OFF,COMP=SYSTCPIP,SUB=(tcpiproc)
TRACE CT,WTRSTOP=ctw
```


The external data set (specified in the procedure "ctw") is now available for formatting.

Filtering options when recording the trace: Options for filtering include the following:

Component

Required - SYSTCPIP for SOCKAPI.

Subtrace

Required - TCP/IP stack name.

Trace option

Highly recommended to limit the tracing to the SOCKAPI option. You can also filter on this option when formatting the trace.

Jobname

Highly recommended for socket applications to limit the trace to one application. You can also filter on this option when formatting the trace.

ASID Highly recommended as an alternative to the job name if the application has already started running (otherwise, the ASID is unknown). You can also filter on this option when formatting the trace.

IP address

Recommended only for certain scenarios (see discussion below). The IP address is a filtering option when formatting the trace.

Port Recommended only for certain scenarios (see discussion below). The port number is a filtering option when formatting the trace.

If trace data for multiple applications is collected in the same data set or in a dump, the trace output should be filtered so that application programmers see only the data for their applications for security reasons.

The IP address and Port options can be used to filter the trace, both when collecting the trace and when formatting the trace. In general, it is best to collect all the application records to avoid having to recreate the problem. Once the records are collected, you can filter the records various ways when formatting the trace.

An example scenario in which you would only want to collect records for one IP address is if there is a problem with a particular remote client, and the local application has many clients. If you tried to record the trace records for all clients, there could be a lot of data and the trace could wrap, thus overwriting older records. Note that if you specify an IP address when collecting the trace, the trace records with no IP address are also collected. So you get all the records for the problem client, and some other client records.

An example scenario, in which you would only want to collect records for one port number, is if there is a problem with a server on one port. If you specify a port number when collecting the trace, the trace records with no port number are also collected. You get all the records for the problem server application, and some other applications' records.

IP address/port filtering, when specified, has a varying effect depending on the type of socket call being traced. Table 12 on page 79 describes the effect of IP address/port filtering for the different types of socket API calls. The Yes or No specified in columns 2 and 3 indicates whether local port filtering and remote IP address filtering can be activated for the socket calls in column 1. Yes means that if a filter is set, only the calls matching that filter are collected. No means that

whether or not a filter is specified, all the calls are collected (no filtering is done).

Table 12. IP address and port filtering effect on different types of socket API calls

Socket call	Filtering active?	
	Local port	Remote IP address
ACCEPT	Yes	No (1)
BIND	Yes/No (2)	No
CONNECT	Yes/No (3)	Yes
CANCEL FREEADDRINF GETADDRINFO GETCLIENTID GETHOSTBYADDR GETHOSTBYNAME GETHOSTID GETHOSTNAME GETNAMEINFO INITAPI RECVFROM RCVMSG SELECT SELECTEX SENDMSG SENDTO SOCKET TAKESOCKET TERMAPI	No	No
LISTEN	Yes	No
CLOSE GETPEERNAME GETSOCKNAME GETSOCKOPT GIVESOCKET FNCTL IOCTL READ READV RECV SHUTDOWN SEND SETSOCKOPT WRITE WRITEV	Yes	Yes

Where Yes is indicated above, the assumption is made that the information necessary for the filtering option is available. For example, if a SEND is issued on a

socket that is not bound or not connected, no filtering takes place. In addition, the following describe some of the special considerations for the different socket calls in the previous table.

1. Even though the remote IP address is available after an ACCEPT call, it is not used for filtering the exit ACCEPT trace record. This is done to avoid confusion where the entry trace record for ACCEPT would not be filtered, but the exit trace record would.
2. Assumes a BIND issued for a nonzero port. If a BIND is issued for port 0 (meaning an ephemeral port is assigned by TCP/IP), no filtering takes place for this BIND call.
3. If the socket is bound at the time of the CONNECT, local port filtering will be honored. Otherwise the CONNECT will not be subject to local port filtering.

Monitoring the trace: Use the MVS command DISPLAY TRACE to check the trace options currently in effect. The following is an example of a console showing the display command and the resulting output (the line numbers were added for discussion reference).

```

1.      14.27.14 D TRACE,COMP=SYSTCPIP,SUB=(tcpiproc)
2.      14.27.14 IEE843I 14.27.14 TRACE DISPLAY
3.              SYSTEM STATUS INFORMATION
4.      ST=(ON,0064K,00064K) AS=ON BR=OFF EX=ON MT=(ON,064K)
5.      TRACENAME
6.      =====
7.      SYSTCPIP
8.
9.              MODE BUFFER HEAD SUBS
10.             =====
11.             OFF          HEAD    1
12.      NO HEAD OPTIONS
13.      SUBTRACE      MODE BUFFER HEAD SUBS
14.      -----
15.      TCPIPROC      ON    0008M
16.      ASIDS         *NONE*
17.      JOBNAME       EZASOKJS
18.      OPTIONS       SOCKAPI
19.      WRITER        CTW

```

For component SYSTCPIP, do not be misled by line 10 in the example. It always says the trace is off because TCP/IP uses the subtrace for all tracing. The subtrace TCPIPROC on line 14 indicates the actual state of the trace. In this example, the trace is active (ON) with an internal buffer size of eight megabytes and only the SOCKAPI option is active. Only one application (EZASOKJS) is being traced and the trace is being written to an external writer.

Line Description

- | | |
|------|--|
| 1 | The MVS DISPLAY TRACE command. For more information on this command, see <i>z/OS MVS System Commands</i> . |
| 2–4 | These are explained in the <i>z/OS MVS System Messages, Vol 1 (ABA-AOM)</i> for IEE843I. |
| 5–7 | Show that this is the CTRACE component SYSTCPIP. |
| 8–11 | These are not applicable for TCP/IP because TCP/IP uses only the subtrace facility of the MVS CTRACE service. Instead of activating a global trace, the trace options are specified for each stack individually. ¹ Thus, there can be multiple TCP/IP stacks with different CTRACE options. Note however that line 10 is useful — it shows that there is one subtrace (meaning one TCP/IP stack is active). |
| 14 | Shows the "subtrace" name is the TCP/IP procedure name (TCPIPROC in |

this example), whether the trace is active (MODE=ON), and the buffer size is eight megabytes. The buffer size is the number of bytes in the data space that is used for recording the trace.

- 15–16** Show the ASID and JOBNAME filtering values. If any ASIDs or JOBNAMEs are listed, only those trace entries matching the ASID or JOBNAME will be collected. "ASIDS *NONE*" indicates that all address spaces are being traced (there is no filtering).
- 17** Shows the specific options that are active, as specified in the TRACE command or in the CTIEZBxx PARMLIB member. If port or IP address filtering were active, they would appear on this line.
- 18** Shows the external writer is inactive. If the writer is active, the writer procedure name is shown instead of *NONE*.

Capturing the trace: If you use only the internal buffer, you must obtain a dump with the TCP/IP data space (TCPIPDS1) in order to view the CTRACE records. It is usually a good idea to also capture the application address space. For example, using the MVS DUMP command, type the following commands. Be sure to specify the TCP/IP data space (TCPIPDS1) because that is where the CTRACE data is located. Note that the SDATA options specified are appended to other options. The SDATA options shown here are the generally recommended options.

```
DUMP COMM=(Sample dump for SOCKAPI)
R n,JOBNAME=(tcpiproc,ezasokjs),DSPNAME=('tcpiproc'.TCPIPDS1),CONT
R n,SDATA=(ALLNUC,CSA,LPA,LSQA,RGN,SWA,SQA,TRT),CONT
R n,END
```

Notes:

1. You can type the first three commands in advance, and you can then just type the fourth command at the correct moment to capture the events.
2. If you use the external writer, "External writer" on page 77, explains how to capture the trace in a data set.

How to format the SOCKAPI trace option

Use the IPCS CTRACE command to format the trace, both for a dump and for an external writer. Interactively, you can either type the CTRACE command on the IPCS Command panel or you can use the panel interface. IPCS is also available in batch. Whichever interface you choose, for TCP/IP we recommend using the CTRACE QUERY command to find out what subtraces are contained in the data set. For example, the command CTRACE QUERY(SYSTCPIP) SHORT produced the following output:

```
COMPONENT TRACE QUERY SUMMARY

      COMPONENT SUB NAME
      -----
0001. SYSTCPIP  TCPSVT
0002. SYSTCPIP  TCPSVT3
0003. SYSTCPIP  TCPSVT1
0004. SYSTCPIP  TCPSVT2
```

There are several filters available that can help to limit the amount of data formatted. In addition to the CTRACE options (start and stop time, and such) provided by IPCS, there are some options specifically for TCP/IP:

DUCB Not applicable for SOCKAPI. (DUCB is an internal TCP/IP token.)

CID (connection identifier)

Not applicable for SOCKAPI.

IPADDR

Can be used for SOCKAPI. Specify the IPv4 addresses in dotted decimal format, with an optional prefix value (1 to 32) or a subnet mask in dotted decimal form. Specify the IPv6 address in colon hexadecimal notation (or in a combination of colon hexadecimal and dotted decimal for IPv4-mapped IPv6 addresses), with an optional prefix value (1 to 128). Several socket calls do not use an IP address. To see the trace records without an IP address (or with an IP address of all zeros), specify zero for one of the IPADDR values. For example IPADDR(0,9.67.113/24) will format all CTRACE records with an IP address of 000.000.000.000 and will format all CTRACE records with an IP address of 009.067.113.*, where * is any number from 0 to 255.

PORT Can be used for SOCKAPI. Specify the port number in decimal. Several socket calls do not have an associated port number, such as INITAPI and SOCKET. To see the trace records without a port (or with a port of 0), specify zero for one of the port values. For example, PORT(0,389,1925).

You can save the formatted output to the IPCSPRNT data set.

If the formatted output does not contain the records you expect:

- In a dump, you can check the options specified when recording the trace by using the TCPIPCS TRACE command to display the TCP/IP CTRACE filtering options in effect. This also indicates whether any records were lost. See Chapter 6, "IPCS subcommands for TCP/IP" on page 169 for more information on the TCPIPCS TRACE command.
- For either a dump or an external writer data set, use the CTRACE QUERY command to see what tracing was in effect (subtrace name, start and stop times). For a dump, this command will also show the buffer size and options. For example, the command CTRACE QUERY(SYSTCPIP) SUB((TCPIPROC)) FULL produced the following output for a dump:

```
COMPONENT TRACE QUERY SUMMARY
```

```
COMP(SYSTCPIP)SUBNAME((TCPIPROC))
```

```
START = 01/10/2000 19:49:21.234490 GMT
```

```
STOP = 01/10/2000 19:51:51.360653
```

```
Buffer size: 0256K
```

```
OPTIONS: ACCESS ,OPCMDS ,OPMSGs ,QUEUE ,ROUTE ,INIT ,SOCKAPI ,SOCKET
```

```
OPTIONS: MINIMUM
```

For TCP/IP, the first line of "options" (showing ACCESS) is the applicable one. This shows the options as specified on the command line or in the CTIEZBxx PARMLIB member.

Refer to the *z/OS MVS IPCS User's Guide* for more information about CTRACE formatting. Refer to *z/OS MVS IPCS Commands* for more information about the CTRACE command.

How to read and interpret the SOCKAPI trace option

The SOCKAPI trace records trace the input and output parameters for most of the API calls. The API calls not traced are GETIBMOPT, TASK, GLOBAL, NTOP, PTON, and any API calls that fail before the trace point is reached. (An API call will fail if module EZBSOH03 cannot be located, if EZBSOH03 is unable to obtain storage, and so on.) In addition to tracing API calls, trace records are created for a few special situations (Default INITAPI and Unsolicited Event exit being driven). For

API calls, there is an Entry record describing the input parameters, and an Exit record describing the output parameters (with some input parameters repeated for clarification). For asynchronous calls, there is also an Async Complete (Asynchronous Complete) record (see “Examples of SOCKAPI trace records” on page 84).

The following examples include:

- A SOCKAPI trace record
- Trace records for asynchronous applications
- Resolver API calls
- External IOCTL commands
- API Call with an IOV parameter
- Default INITAPI
- Default TERMAPI
- SELECT
- SELECTEX
- Token error
- Unsolicited event exit

A SOCKAPI trace record: A typical SOCKAPI record is shown below. This example is a READ Entry.

The lines are numbered for discussion reference only. The description for each line is for the example shown. Lines 1-5 are the separator and header lines that exist for all SOCKAPI trace records. Lines 6-7 are optional header lines.

The parameters for the specific call follow the header lines. For Entry records, the input parameters are shown. For Exit and Asynchronous Complete records, the output parameters are shown and some input parameters may also be shown for reference. Parameters are only formatted if they were specified in the call (optional parameters not supplied are not formatted). The parameters are listed in a specific order for consistency. The parameter names are the same as the names in the *z/OS Communications Server: IP Application Programming Interface Guide* with a few exceptions; for example, S is formatted as SOCKET. The parameter name, value, and address are shown on one line if the value fits. Numeric parameter values are in decimal unless followed by a lowercase **x** indicating hexadecimal. Whenever possible, the values are interpreted (such as ERRNO) for reference.

```

1. =====00007FE8
2. MVS026 SOCKAPI 60050042 19:31:08.338135 READ Entry
3. HASID....0027 PASID....0027 SASID..0027 JOBNAME..EZASOKGS
4. TCB.....006E6A68 TIE.....00008DF8 PLIST..00008E0C DUCB....0000000C KEY..8
5. ADSNAME..GTASOKGS SUBTASK..MACROGIV TOKEN....7F6F3798 09902FB0
6. LOCAL PORT..12035 IPADDR.. 9.67.113.58
7. REMOTE PORT..1034 IPADDR.. F901::32E1
8. REQAREA... 00008D90x Addr..00008D90
9. SOCKET....: 1 Addr..00008A38
10. NBYTE.....: 40 Addr..00008A34
11. ALET.....: 00000000x Addr..000089A8
12. BUF.....: (NO DATA) Addr..000089A8

```

Line Description

1 This separator line shows the previous SYSTCPIP component trace record number in hexadecimal.

- 2 The first data line has the host name (MVS026), trace option (SOCKAPI), trace code (60050042), time, and trace record name.
- 3 The home, primary, and secondary ASIDs are always the same value (application's ASID) for the SOCKAPI trace option. The job name is also shown.
- 4 The MVS TCB address is shown. TIE (Task Interface Element) is the value of the TASK parameter on the EZASMI macro. The TIE is described in the *z/OS Communications Server: IP Application Programming Interface Guide*. The parameter list address and DUCB are shown. Multiple concurrent calls can use the TIE; if so, they must have a different PLIST. The key is the 4-bit storage key from the PSW.
- 5 The ADSNAME (from the INITAPI call) is formatted in EBCDIC. The subtask name (from the INITAPI call) is formatted in EBCDIC if possible; otherwise, it is formatted in hexadecimal. The token is an eight-byte value, which identifies the INITAPI call instance.
- 6–7 If applicable, the ports and IP addresses are shown. The ports are formatted in decimal; the IP addresses are in dotted decimal.
- 8 The REQAREA parameter is shown because it was specified by the application. This is the 4-byte token presented to the application's exit when the response to the function request is complete. At the far right, the address in the application program of the REQAREA parameter is shown.
- 9 The SOCKET parameter is formatted in decimal. Its address is also shown.
- 10 The NBYTE parameter (number of bytes to be read) is formatted in decimal, followed by its address.
- 11 The ALET parameter is formatted in hexadecimal, followed by its address.
- 12 The BUF parameter currently has no data (because no data has been read) but its address is shown. In the READ Exit (or READ Async Complete) record, if the call was successful, the first 96 bytes of the data are also shown.

Examples of SOCKAPI trace records:

Successful API Call: For asynchronous APIs, the Exit record merely indicates whether or not the call was acceptable. The contents of general purpose register 15 are displayed to indicate this. The Asynchronous Complete record shows the actual results of the call. In addition to the output parameters, several interesting values are traced, including the contents of general purpose register 0, the pointer to the asynchronous exit routine, the token passed to the asynchronous exit, the key in which the asynchronous exit was invoked, and the authorization state in which the exit is invoked. These values are not parameters on the GETHOSTID call, so their addresses are not shown. In this example, note also that the return code is formatted in dotted decimal and the meaning of the return code is provided.

Note: The API call may actually complete synchronously, in which case the Async Complete trace record may appear in the trace prior to the Exit record.

```
=====00007B01
MVS026  SOCKAPI  60050012  19:27:08.111729  GETHOSTID Exit
HASID...0027  PASID...0027  SASID..0027  JOBNAME..EZASOKOS
TCB.....006E6A68  TIE.....00006DF8  PLIST..00006E0C  DUCB.....0000000C  KEY..8
ADSNAME..EZASOKOS  SUBTASK..00000000  00000000  TOKEN...7F6F3798  09902FB0
REQAREA...: 00006D90x  Addr..00006D90
R15.....: 0 (CALL ACCEPTED)
=====00007B05
```

```

MVS026  SOCKAPI  60050032  19:27:08.111741  GETHOSTID Async Complete
HASID....0027    PASID....0027    SASID..0027    JOBNAME..EZASOKOS
TCB.....006E6A68 TIE.....00006DF8 PLIST..00006E0C DUCB.....0000000C KEY..8
ADSNAME..EZASOKOS SUBTASK..00000000 00000000    TOKEN....7F6F3798 09902FB0
REQAREA...: 00006D90x                               Addr..00006D90
R0.....: 0x (NORMAL RETURN)
ASYNC PTR: 00006B1C
EXIT TOKEN: 00006B98x
EXIT KEY.: 8x
AUTHORIZATION STATE: PROBLEM
RETCODE...: 9.67.113.58      (HOST IP ADDRESS)                               Addr..00006EB4

```

API call fails synchronously: An asynchronous API call may fail synchronously or asynchronously. In this example, the WRITE call error was detected in the synchronous processing, so general purpose register 15 has a nonzero value. The ERRNO value is interpreted (in this case, the NBYTE parameter on the WRITE call had a value of zero, which is not acceptable).

Note: The ERRNO value is the TCP/IP Sockets Extended Return Code. Refer to *z/OS Communications Server: IP and SNA Codes* for information about TCP/IP Sockets Extended Return Codes.

```

=====00007B93
MVS026  SOCKAPI  60050057  19:27:13.817195  WRITE Exit
HASID....0027    PASID....0027    SASID..0027    JOBNAME..EZASOKOS
TCB.....006E6A68 TIE.....00006DF8 PLIST..00006E0C DUCB.....00000009 KEY..8
ADSNAME..EZASOKOS SUBTASK..00000000 00000000    TOKEN....7F6F3798 09902FB0
LOCAL  PORT..11007    IPADDR.. 9.67.113.58
REMOTE PORT..1031    IPADDR.. 9.67.113.58
REQAREA...: 00006D90x                               Addr..00006D90
SOCKET...: 1                                           Addr..00006BDC
R15.....: NON-ZERO (CALL WAS NOT ACCEPTED)
ERRNO....: 10184 (EIBMWRITELZERO)                     Addr..00006EB0
RETCODE...: -1                                         Addr..00006EB4

```

API call fails synchronously with parameter not addressable: If a parameter specified in the API call is not addressable by TCP/IP when creating the SOCKAPI record, the string (** PARAMETER NOT ADDRESSABLE **) is shown instead of the parameter value. The parameter address is shown at the far right, as usual.

```

=====00021347A
VIC102  SOCKAPI  60050050  17:36:51.302111  SEND Entry
HASID....0026    PASID....0026    SASID..0026    JOBNAME..USER2
TCB.....006D6D50 TIE.....0000BDF8 PLIST..0000BE0C DUCB.....00000009 KEY..8
ADSNAME..USER2   SUBTASK..EZASOKEC    TOKEN....7F755798 09806FB0
LOCAL  PORT..0    IPADDR ..0.0.0.0
REMOTE PORT..11007    IPADDR ..9.37.65.134
SOCKET...: 0                                           Addr..0000BA50
NBYTE...: 96                                           Addr..0000BA6C
BUF.....: (** PARAMETER NOT ADDRESSABLE **)           Addr..00015F38
FLAGS....: 0 (NONE)                                   Addr..0000BC04

```

API call fails synchronously with diagnostic reason code: If the API call does not complete successfully, the return code, ERRNO value (in decimal and interpreted), and possibly a diagnostic reason code are shown. The first two bytes of the diagnostic reason code are a qualifier (IBM internal use only). The last two bytes of the diagnostic reason codes are the UNIX ERRNOJR values described in the *z/OS Communications Server: IP and SNA Codes*.

```

=====000085C1
MVS026  SOCKAPI  60050004  19:36:01.934828  ACCEPT Exit
HASID....01F6    PASID....01F6    SASID..01F6    JOBNAME..EZASOKUE
TCB.....006E6A68 TIE.....00006DF0 PLIST..00006E04 DUCB.....0000000D KEY..8
ADSNAME..EZASOKUE SUBTASK..EZASOKUE    TOKEN....7F6F3798 09902FB0

```



```

LOCAL PORT..11007      IPADDR ..9.67.113.58
REMOTE PORT..0         IPADDR ..0.0.0.0
REQAREA...: 00000000x      Addr..00006D80
SOCKET....: 0              Addr..00006BA8
NAME.....: (NO DATA)      Addr..00006BAC
DIAG. RSN: 76620291x
ERRNO.....: 5 (EIO)        Addr..00006EA8
RETCODE...: -1              Addr..00006EAC

```

Resolver API calls: The GETHOSTBYADDR and GETHOSTBYNAME IPv4 Resolver API calls use the HOSTENT structure described in the calls in the *z/OS Communications Server: IP Application Programming Interface Guide*. As shown in the following GETHOSTBYADDR Exit trace example,, the HOSTENT address is shown on one line, and the contents of the HOSTENT structure are described on separate lines. There can be multiple aliases and host addresses; each one is listed separately. In this example, there are two aliases.

```

=====000051CB
MVS026  SOCKAPI  60050066  19:02:01.426345  GETHOSTBYADDR Exit

HASID....0027      PASID....0027      SASID..0027      JOBNAME..EZASOKGH
TCB.....006E6A68  TIE.....00007DF8  PLIST..00007E0C  DUCB....0000000A  KEY..0
ADSNAM...EZASOKGH  SUBTASK..00000000  00000000      TOKEN....00000000  09902FB0
HOSTENT...:                      Addr..00005F08
HOSTNAME...:                      Addr..00005F30
Loopback
FAMILY....: 2                      Addr..00005F10
ADDR LEN...: 4                      Addr..00005F14
HOSTADDR...: 127.0.0.1              Addr..00005F54
ALIAS.....: LOOPBACK                Addr..00005F3C
ALIAS.....: LOCALHOST                Addr..00005F48
RETCODE...: 0                      Addr..00007EB4

```

The GETADDRINFO for IPv4 or IPv6 Resolver API shows the call is requesting the IP address for the host (node) name MVS150. No service name is provided. GETADDRINFO exit shows the hostname was resolved to the IPv4 address 9.67.113.117. These fields are described in the Macro and CALL section in the *z/OS Communications Server: IP Application Programming Interface Guide*.

```

=====0000134C
MVS150  SOCKAPI  6005006D  15:06:07.294268  GETADDRINFO Entry
HASID....002D      PASID....002D      SASID..002D      JOBNAME..USER1X
TCB.....007F63B0  TIE.....0A90AAD8  PLIST..0A90AAEC  DUCB....00000009  KEY..8
ADSNAM...SUBTASK..EZAS06CS      TOKEN....7F694220  0A97EFB0
NODELEN...: 6                      Addr..0A973490
NODE.....:                      Addr..0A973390
MVS150
SERVLEN...: 0                      Addr..0A9734B8
SERVICE...: (NO DATA)              Addr..0A973498
HINTS.....: 0A913F70x (ADDRINFO Address)  Addr..0A913F90
ADDRINFO Structure...:
AF.....: 0 (AF_UNSPEC)      FLAGS..... 00000002x
SOCTYPE..: 0 (UNKNOWN)      PROTO..... 0 (IPPROTO_IP)
NAME.....: 00000000x      NAMELEN... 0
CANONNAME 00000000x      NEXT..... 00000000x
CANNLEN...: (NO DATA)                      Addr..0A9734C0
RES.....: (NO DATA)                      Addr..0A913F94
=====0000134D
MVS150  SOCKAPI  6005006E  15:06:09.997756  GETADDRINFO Exit
HASID....002D      PASID....002D      SASID..002D      JOBNAME..USER1X
TCB.....007F63B0  TIE.....0A90AAD8  PLIST..0A90AAEC  DUCB....00000009  KEY..8
ADSNAM...SUBTASK..EZAS06CS      TOKEN....7F694220  0A97EFB0
HINTS.....: 0A913F70x (ADDRINFO Address)  Addr..0A913F90
ADDRINFO Structure...:
AF.....: 0 (AF_UNSPEC)      FLAGS..... 00000002x
SOCTYPE..: 0 (UNKNOWN)      PROTO..... 0 (IPPROTO_IP)

```



```

NAME..... 0002111Cx      NAMELEN... 0
PORT..... 0              IPADDR.... 0.0.0.0
FAMILY.. 0 (UNKNOWN)     RESERVED.. 0000000000000000x
CANONNAME 00000000x      NEXT..... 00000000x
CANNLEN..: 22                      Addr..0A9734C0
RES.....: 0002111Cx (ADDRINFO Address) Addr..0A913F94
ADDRINFO Structure...:
AF..... 2 (AF_INET)      FLAGS..... 00000000x
SOCTYPE.. 1 (STREAM)     PROTO..... 0 (IPPROTO_IP)
NAME..... 0002114Cx      NAMELEN... 16
PORT.... 0              IPADDR.... 9.67.113.117
FAMILY.. 2 (AF_INET)     RESERVED.. 0000000000000000x
CANONNAME 0002101Cx      NEXT..... 00000000x
MVS150.raleigh.ibm.com

```

The FREEADDRINFO for IPv4 or IPv6 Resolver API call displays the RES (ADDRINFO) structure that will be freed. This field is in the Macro and CALL section in the *z/OS Communications Server: IP Application Programming Interface Guide*.

```

=====0000134E
MVS150  SOCKAPI  6005006F  15:06:09.998002  FREEADDRINFO Entry
HASID....002D   PASID....002D   SASID..002D   JOBNAME..USER1X
TCB.....007F63B0 TIE.....0A90AAD8 PLIST..0A90AAEC DUCB.....00000009 KEY..8
ADSDNAME..... SUBTASK..EZAS06CS      TOKEN....7F694220 0A97EFB0
ADDRINFO.: 0002111Cx (ADDRINFO Address) Addr..0A913F94
ADDRINFO Structure...:
AF..... 2 (AF_INET)      FLAGS..... 00000000x
SOCTYPE.. 1 (STREAM)     PROTO..... 0 (IPPROTO_IP)
NAME..... 0002114Cx      NAMELEN... 16
PORT.... 0              IPADDR.... 9.67.113.117
FAMILY.. 2 (AF_INET)     RESERVED.. 0000000000000000x
CANONNAME 0002101Cx      NEXT..... 00000000x
MVS150.raleigh.ibm.com
=====0000134F
MVS150  SOCKAPI  60050070  15:06:09.999021  FREEADDRINFO Exit
HASID....002D   PASID....002D   SASID..002D   JOBNAME..USER1X
TCB.....007F63B0 TIE.....0A90AAD8 PLIST..0A90AAEC DUCB.....00000009 KEY..8
ADSDNAME..... SUBTASK..EZAS06CS      TOKEN....7F694220 0A97EFB0

```

The GETNAMEINFO for IPv4 or IPv6 Resolver API shows the call is requesting the name of the IPv6 address ::1 and the service name for port 1031. GETNAMEINFO Exit shows the IP address was resolved to the name loop6int.resdns.ibm.com and no service name was found for port 1031 (hence the service name is the input port number). These fields are in the Macro and CALL section in the *z/OS Communications Server: IP Application Programming Interface Guide*.

```

=====0000135F
MVS150  SOCKAPI  6005006B  15:06:45.481639  GETNAMEINFO Entry
HASID....0025   PASID....0025   SASID..0025   JOBNAME..USER1Y
TCB.....007F62F8 TIE.....0A903AD8 PLIST..0A903AEC DUCB.....00000008 KEY..8
ADSDNAME..... SUBTASK..EZS06CC      TOKEN....7F6E2220 0A977FB0
NAMELEN..: 28                      Addr..0A96C348
NAME.....:                          Addr..0A96C500
PORT... 1031      IPADDR.. ::1
FAMILY.. 19 (AF_INET6) LENGTH.. 0
FLOWINFO. 00000000x SCOPID.. 00000000x
HOSTLEN..: 255                      Addr..0A96C450
HOST.....: (NO DATA)              Addr..0A96C350
SERVLEN..: 32                      Addr..0A96C478
SERVICE..: (NO DATA)            Addr..0A96C458
FLAGS.....: 00000004x              Addr..0A96C480
=====00001360
MVS150  SOCKAPI  6005006C  15:06:46.707053  GETNAMEINFO Exit
HASID....0025   PASID....0025   SASID..0025   JOBNAME..USER1Y
TCB.....007F62F8 TIE.....0A903AD8 PLIST..0A903AEC DUCB.....00000008 KEY..8

```

```

| ADSNAME..... SUBTASK..EZS06CC          TOKEN....7F6E2220 0A977FB0
| HOSTLEN...: 23                          Addr..0A96C450
| HOST.....:                             Addr..0A96C350
|      loop6int.resdns.ibm.com
| SERVLN...: 4                            Addr..0A96C478
| SERVICE...: 1031                        Addr..0A96C458
| FLAGS....: 00000004x                   Addr..0A96C480

```

External IOCTL commands: For external IOCTL commands, the command name is interpreted. For IBM internal-use-only commands, the hexadecimal value of the command is shown. The input and output for each command can differ. In this example, the SIOCGIFCONF command requests the network interface configuration. The exit record shows the call was successful (the return code is zero) and the network interface configuration is shown.

```

=====00001734
MVS026  SOCKAPI  6005001F  20:42:44.805938  IOCTL Entry

HASID....19      PASID....19      SASID..19      JOBNAME..USER1
TCB.....006AFD40 TIE.....68DF8  PLIST..00068E0C DUCB.....00000008 KEY..8
ADSNAME..USER1  SUBTASK..00000000 00000000  TOKEN....7F67F798 0A2B4FB0
LOCAL  PORT..11007  IPADDR ..9.67.113.58
REMOTE PORT..0      IPADDR ..0.0.0.0
SOCKET...: 0                          Addr..000685A0
COMMAND...: SIOCGIFCONF               Addr..0006782C
REQARG...:                             Addr..00068928
BUFFER LENGTH.. 99

=====00000323
MVS026  SOCKAPI  60050020  20:42:44.806101  IOCTL Exit

HASID....19      PASID....19      SASID..19      JOBNAME..USER1
TCB.....006AFD40 TIE.....68DF8  PLIST..00068E0C DUCB.....00000008 KEY..8
ADSNAME..USER1  SUBTASK..00000000 00000000  TOKEN....7F67F798 0A2B4FB0
LOCAL  PORT..11007  IPADDR ..9.67.113.58
REMOTE PORT..0      IPADDR ..0.0.0.0
SOCKET...: 0                          Addr..000685A0
COMMAND...: SIOCGIFCONF               Addr..0006782C
RETARG...:                             Addr..000685C4
Socket Name.. TR1
PORT.... 0                          IPADDR.... 9.67.113.58
FAMILY.. 2 (AF_INET)  RESERVED.. 0000000000000000x
RETCODE...: 0                          Addr..00068EB4

```

API call with an IOV parameter: The IOV parameter is an array of structures used on the READV, RECVMSG, SENDMSG, and WRITEV API calls. Each structure contains three words: the buffer address, the ALET, and the buffer length. Each IOV entry is shown on one line. When there is data available (READV Exit, RECVMSG Exit, SENDMSG Entry, and WRITEV Entry), some of the buffer data is also displayed. A maximum of 96 bytes of data are displayed.

In the READV Exit example, three IOV entries were specified, but only two were used. All the data is displayed because the total is less than 96 bytes.

```

=====00001773
MVS026  SOCKAPI  60050045  19:19:20.954789  READV Exit

HASID....0024    PASID....0024    SASID..0024    JOBNAME..EZASOKKS
TCB.....006E6A68 TIE.....00007DF8 PLIST..00007E0C DUCB.....0000000B KEY..8
ADSNAME..EZASOKKS SUBTASK..EZASOKKS  TOKEN....7F6F3798 09902FB0
LOCAL  PORT..11007  IPADDR ..9.67.113.58
REMOTE PORT..1032  IPADDR ..9.67.113.58
REQAREA...: 00007D90x                Addr..00007D90
SOCKET...: 1                          Addr..0000776C
IOVCNT...: 3                          Addr..000077B4
IOENTRY..: LENGTH..10  ALET..0x        Addr..00007890

```

+0000	E38889A2	4089A240	8396	This is co	
IOENTRY.: LENGTH..10	ALET..0x			Addr..0000789A	
+0000	99998583	A34B		rrect.	
IOENTRY.: LENGTH..10	ALET..0x			Addr..000078A4	
RETCODE... 16 BYTES TRANSFERRED				Addr..00007EB4	

Default INITAPI: An explicit INITAPI call is not required prior to some API calls, so TCP/IP creates a default INITAPI. (Refer to the *z/OS Communications Server: IP Application Programming Interface Guide* for the complete list.) The default INITAPI record is traced after the Entry record for the API call that caused the default INITAPI to occur. There is just one record for this event (no Exit record).

```
=====000077EC
MVS026  SOCKAPI  60050040  19:24:11.552924  Default INITAPI

HASID....0027      PASID....0027      SASID..0027      JOBNAME..EZASOKSX
TCB.....006E6A68  TIE.....00007DF0  PLIST..00007E04  DUCB.....0000000A  KEY..8
ADSNAM..EZASOKSX  SUBTASK..00000000  00000000      TOKEN....7F6F3798  09902FB0
MAXSNO...: 49
APITYPE...: 2
RETCODE...: 0
```

Default TERMAPI: Usually, an application will end the connection between itself and TCP/IP by issuing the TERMAPI call. But sometimes, the connection ends for another reason, such as the application being cancelled. In this case, TCP/IP will issue a default TERMAPI. The default TERMAPI is traced in a SOCKAPI trace record. There is just one record for this event (no Exit record).

```
=====00000168
MVS026  SOCKAPI  60050069  22:46:48.185419  Default TERMAPI

HASID....01F9      PASID....01F9      SASID..01F9      JOBNAME..EZASOKQS
TCB.....006E6A68  TIE.....08920888  PLIST..00000000  DUCB.....00000008  KEY..6
ADSNAM..EZASOKQS  SUBTASK..EZASOKQS      TOKEN....7F6F3798  00000000
```

SELECT: For SELECT and SELECTEX, the socket masks are formatted in both binary and decimal. The socket list is displayed first in binary. The socket numbers are indicated by the bit position in the mask, starting with bit position 0 (for socket 0), which is the far right bit. The bit positions (socket numbers) are shown at left. For example, the lowest numbered sockets are on the last line; they are sockets 0 to 31. In this line, only sockets 0, 1, 2, and 3 are selected. Below the binary mask, the decimal socket numbers are listed in numerical order. This is a very handy way to check whether the mask is coded as it is intended.

```
=====00024EDF
BOTSWANA SOCKAPI  6005004C  20:51:35.477605  SELECT Entry

HASID....0078      PASID....0078      SASID..0078      JOBNAME..TN1
TCB.....007F6988  TIE.....1463227C  PLIST..1477EF18  DUCB.....00000016  KEY..8
ADSNAM..          SUBTASK..14632138      TOKEN....7F75FFC8  1468FA90
REQAREA...: 1477EEF0x                      Addr..1477EF98
MAXSOC...: 100                      Addr..14632258
TIMEOUT...: SECOND..0      MICRO SECOND..500000      Addr..1463226C
RSNDMSK...:                      Addr..14632108
SOCKET NO.      READ SOCKET MASK (INPUT)
(Decimal)      (Binary)
127  96                      11110111
95   64      11111111 11111111 10111111 11111111
63   32      00111011 11111111 11111111 11111101
31    0      00000000 00000000 00000000 00001111
SELECTED SOCKETS:
0,   1,   2,   3,   32,  34,  35,  36,  37,  38
39,  40,  41,  42,  43,  44,  45,  46,  47,  48
49,  50,  51,  52,  53,  54,  55,  56,  57,  59
60,  61,  64,  65,  66,  67,  68,  69,  70,  71
```

```

72, 73, 74, 75, 76, 77, 79, 80, 81, 82
83, 84, 85, 86, 87, 88, 89, 90, 91, 92
93, 94, 95, 96, 97, 98, 100, 101, 102, 103

```

SELECTEX: The SELECTEX call can contain a list of ECBs. The high-order bit on the SELECB address indicates whether or not a list of ECBs was specified. Since the high-order bit is on in this example, there is a list of ECBs. The end of the list is indicated by the high-order bit in the ECB address. In this example, the time limit expired before any ECBs were posted. Since no selected sockets were ready, the read, write, and error masks indicate there is no data to report.

```

=====000078FB
MVS026  SOCKAPI  6005004F  19:25:48.610379  SELECTEX Exit

HASID....0027      PASID....0027      SASID..0027      JOBNAME..EZASOKX4
TCB.....006E6A68  TIE.....00007DF8  PLIST..00007E0C  DUCB.....0000000C  KEY..8
ADSNAME..EZASOKX4  SUBTASK..BARBARA                TOKEN....7F6F3798 09902FB0
MAXSOC...: 33                      Addr..00007AE8
TIMEOUT...: SECOND..0             MICRO SECOND..35      Addr..00007AF4
RRETSK...: (NO DATA)              Addr..00007B0C
WRETSK...: (NO DATA)              Addr..00007B14
ERETS...: (NO DATA)              Addr..00007B1C
SELECB...:                      Addr..80007B60
ECB.....: 00000000x              Addr..00007B70
ECB.....: 00000000x              Addr..00007B74
ECB.....: 00000000x              Addr..00007B78
ECB.....: 00000000x              Addr..80007B7C
RETCODE...: 0 (TIME LIMIT EXPIRED) Addr..00007EB4

```

Token Error: When an API call fails very early in processing, before the SOCKAPI Entry record is created, the Token Error SOCKAPI record is written. In the example, the BIND call failed due to the token being overwritten (the token at offset eight has X'FFFF'). There is no BIND Entry or Exit record.

```

=====00000158
MVS026  SOCKAPI  6005006A  22:46:48.173348  Token Error

HASID....01F9      PASID....01F9      SASID..01F9      JOBNAME..EZASOKQS
TCB.....006E6A68  TIE.....00006DF8  PLIST..00006E0C  DUCB.....00000008  KEY..8
ADSNAME..          SUBTASK..          TOKEN....7F6F3798 09902FB0
CALL.....: BIND
TOKEN....: 7F6F3798 09902FB0 FFFF0000 00003FC5x
ERRNO....: 1028 (EIBMINVTCPCONNECTION)
RETCODE...: -1

```

Unsolicited event exit: If the unsolicited event exit is driven, a SOCKAPI trace record will be created (if the SOCKAPI trace option is active).

Note: The key in the header is 0. This means the UEE trace record was created when TCP/IP was in key zero. The UEEXIT has key 8, which means the UE exit will be invoked in key eight.

```

=====000086FC
MVS026  SOCKAPI  60050041  19:36:04.965468  Unsolicited Event Exit Invoked

HASID....0024      PASID....0024      SASID..0024      JOBNAME..TCPIPROC
TCB.....006E6A40  TIE.....00006DF0  PLIST..00000000  DUCB.....00000000  KEY..0
ADSNAME..EZASOKUE  SUBTASK..EZASOKUE                TOKEN....7F6F3798 00000000
UEEXIT...: ADDRESS..00006B30  TOKEN..00006D80x  ASCB.....00F94C80x  KEY..8
          REASON...1 (TCP/IP TERMINATION)

```

How to correlate the data trace and packet trace with the SOCKAPI trace

The SOCKAPI option only records the first 96 bytes of data. To see all the data that was sent or received, you must also activate the data trace or packet trace. The

data trace can be correlated easily with the SOCKAPI trace option because both traces are recording data between the application and the TCP/IP stack. The traces can be merged with the IPCS MERGE subcommand. The data trace header contains fields that allow the full data to be correlated.

Figure 9 on page 91 shows the data trace record corresponding to the READ Exit SOCKAPI trace entry in Figure 10 on page 92. The server issues READ and waits for a message. The data trace record shows the entire 120 bytes of data because the FULL option was used when starting the data trace. In the READ Exit record, only the first 96 bytes of data are shown. The records in the two traces can be correlated by time, job name, ASID, MVS TCB address, data length, port, and IP address:

Time The data trace time must be prior to the READ Exit record time. The data trace time is 20:08:09.181239. The READ Exit record time is 20:08:09.181354.

Jobname

The job name is EZASOKAS in both records.

ASID The ASID is the server's 0024 (hexadecimal) in both records.

TCB The TCB is 006E6A68 in both records.

Data length

In the data trace, the length is 78 hexadecimal, which is 120 decimal. The SOCKAPI trace record shows that the return code is 120 (decimal) bytes.

Port The source port number in the data trace record (11007 decimal) matches the local port number in the SOCKAPI trace record. The destination and remote ports also match (1040 decimal).

IP Address

The IP addresses are handled in the same way as the port numbers. In this example, both the client and server were on the same TCP/IP stack, so the IP addresses are the same.

```
MVS026    DATA      00000003  20:08:09.181239  Data Trace

JOBNAME =    EZASOKAS          FROM  FULL
TOD CLOCK = XB395B2C2  40035C03
PKT 2          LOST RECORDS = 0      HDR SEQUENCE NUM = 1
SOURCE IP ADDR = 9.67.113.58      DEST IP ADDR = 9.67.113.58
SOURCE PORT = 11007  DEST PORT = 1040  ASID = X0024  TCB = X006E6A68
DATA LENGTH = X0078
0000 E38889A2 4089A240 8140A2A3 99899587 *This is a string|.....@..@.....*
0010 40A689A3 88408696 99A3A840 83888199 * with forty char|@.....@.....*
0020 8183A385 99A24B40 E38889A2 4089A240 *acters. This is |.....K@.....@*
0030 8140A2A3 99899587 40A689A3 88408696 *a string with fo|.@.....@.....*
0040 99A3A840 83888199 8183A385 99A24B40 *rty characters. |...@.....K@*
0050 E38889A2 4089A240 8140A2A3 99899587 *This is a string|.....@..@.....*
0060 40A689A3 88408696 99A3A840 83888199 * with forty char|@.....@.....*
0070 8183A385 99A24B40          *acters.          |.....K@      *
```

Figure 9. Data trace record.

```

=====00002403
MVS026   SOCKAPI   60050043  20:08:09.181354  READ Exit

HASID....0024      PASID....0024      SASID..0024      JOBNAME..EZASOKAS
TCB.....006E6A68  TIE.....00006DF8  PLIST..00006E0C  DUCB.....00000009  KEY..8
ADSNAM...EZASOKAS  SUBTASK..EZASOKAS      TOKEN....7F6F3798  09902FB0
LOCAL  PORT..11007      IPADDR  ..9.67.113.58
REMOTE PORT..1040      IPADDR  ..9.67.113.58
REQAREA...: 00006D90x                      Addr..00006D90
SOCKET....: 1                      Addr..00006B94
NBYTE....: 120                      Addr..00006B90
BUF.....:                      Addr..00006B96
+0000  E38889A2  4089A240  8140A2A3  99899587  | This is a string
+0010  40A689A3  88408696  99A3A840  83888199  |   with forty char
+0020  8183A385  99A24B40  E38889A2  4089A240  |   acters. This is
+0030  8140A2A3  99899587  40A689A3  88408696  |   a string with fo
+0040  99A3A840  83888199  8183A385  99A24B40  |   rty characters.
+0050  E38889A2  4089A240  8140A2A3  99899587  |   This is a string
RETCODE...: 120 BYTES TRANSFERRED          Addr..00006EB4
=====00002407

```

Figure 10. SOCKAPI trace record.

The packet trace, on the other hand, does not correlate well with the SOCKAPI trace option. The packet trace records data being sent or received between the TCP/IP stack and the network. The packet trace data has headers and the data can be segmented or packed.

Packet trace (SYSTCPDA) for TCP/IP stacks

Packet trace is a diagnostic method for obtaining traces of IP packets flowing to and from a TCP/IP stack on a z/OS CS host. You can use the PKTTRACE statement to copy IP packets as they enter or leave TCP/IP, and then examine the contents of the copied packets. To be traced, an IP packet must meet all the conditions specified on the PKTTRACE statement.

The trace process

Trace data is collected as IP packets enter or leave TCP/IP. The actual collection occurs within the device drivers of TCP/IP, which capture the data that has just been received from or sent to the network.

Note: Packets that are captured have extra information added to them before they are stored. This extra information is used during the formatting of the packets. The captured data reflects exactly what the network sees. For example, the trace contains the constituent packets of a fragmented packet exactly as they are received or sent.

The selection criteria for choosing packets to trace are specified through the PKTTRACE statement for the TCP/IP address space. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the PKTTRACE statement and subcommand.

The PKTTRACE statement and subcommand are applied to device links that are defined in the TCP/IP address space through the LINK statement. Figure 11 on page 93 illustrates the overall control and data flow in the IP packet tracing facility.

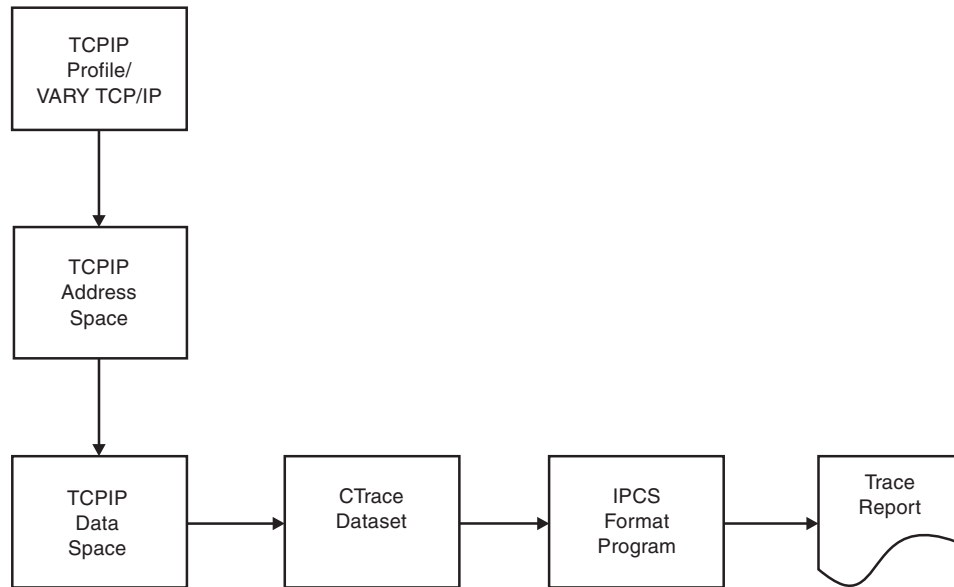


Figure 11. Control and data flow in the IP packet tracing facility

Supported devices

IP packet tracing is supported for all network interfaces supported by TCP/IP (including loopback). However, packets sent and received locally on IP addresses in the PROFILE.TCPIP HOME list will not be traced.

When using the MULTIPATH option of the IPCONFIG statement, packets may be sent over multiple interfaces. All of the interfaces must be traced. In this case specify an IP address to select the required packet. This statement also applies to the case where packets may be received over multiple interfaces (even if MULTIPATH is not used by this TCPIP).

For information about the format of the packet trace command (VARY PKTTRACE) see the *z/OS Communications Server: IP System Administrator's Commands*.

Starting packet trace

To start packet trace, use the following command:

```
V TCPIP,tcpprocname,PKT
```

Security Note: To use any VARY command, the user must be authorized in RACF.

The RACF profile for each user must have access for a resource of the form MVS.VARY.TCPIP.xxx, where xxx is the first eight characters of the command name. For packet trace, this would be MVS.VARY.TCPIP.PKTTRACE.

Traces are placed in an internal buffer, which can then be written out using an external writer. See Chapter 5, "TCP/IP services traces and IPICS support" on page 53. The MVS TRACE command must also be issued for component SYSTCPDA to activate the packet trace.

Once you have started packet trace, you can display the status using the `onetstat` command, as shown in the following example:

```
onetstat -p TCPCS -d
MVS TCP/IP onetstat CS V1R4          TCPIP Name: TCPCS          18:03:31
DevName: LOOPBACK                    DevType: LOOPBACK
DevStatus: Ready
LnkName: LOOPBACK                    LnkType: LOOPBACK    LnkStatus: Ready
  NetNum: 0   QueSize: 0
  BytesIn: 192537                      BytesOut: 192537
  ActMtu: 65535
BSD Routing Parameters:
  MTU Size: 00000                      Metric: 00
  DestAddr: 0.0.0.0                    SubnetMask: 0.0.0.0
Packet Trace Setting:
  Protocol: *                          TrRecCnt: 00000000    PckLength: FULL
  SrcPort: *                          DestPort: *
  IpAddr: 9.67.113.1                  SubNet: *
Multicast Specific:
  Multicast Capability: No
```

In this example, the packet length (PckLength) is FULL and TrRecCnt is the number of packets written for this device.

Note: If you are a TSO user, use the corresponding NETSTAT DEV command.

Modifying options with VARY

After starting a packet trace, you can change the trace using the `VARY` command. For example, if you want to change the packet trace to abbreviate the data being traced, use the following command:

```
V TCPIP,tcpproc,PKT,ABBREV
```

You can display the results of the `VARY` command using `onetstat`:

```
onetstat -p TCPCS -d
MVS TCP/IP onetstat CS V1R4          TCPIP Name: TCPCS          18:17:48
DevName: LOOPBACK                    DevType: LOOPBACK
DevStatus: Ready
LnkName: LOOPBACK                    LnkType: LOOPBACK    LnkStatus: Ready
  NetNum: 0   QueSize: 0
  BytesIn: 813                          BytesOut: 813
  ActMtu: 65535
BSD Routing Parameters:
  MTU Size: 00000                      Metric: 00
  DestAddr: 0.0.0.0                    SubnetMask: 0.0.0.0
Packet Trace Setting:
  Protocol: *                          TrRecCnt: 00000000    PckLength: 00200
  SrcPort: *                          DestPort: *
  IpAddr: *                            SubNet: *
Multicast Specific:
  Multicast Capability: No
```

Note: If you are a TSO user, use the corresponding NETSTAT option.

By issuing multiple `VARY` commands, you can OR filters together. For example, issuing the following `VARY` commands records all packets whose destination port is xxxx or whose source port is xxxx.

```
V TCPIP,tcpproc,PKTTRACE,DEST=xxxx
V TCPIP,tcpproc,PKTTRACE,SRCP=xxxx
```


The result is a trace that contains only packets with a source port of xxxx or packets with a destination port of xxxx.

If both DEST and SRCP are specified in the same command, you can AND the parameters together. For example, issuing the following VARY command records only the packets with both a destination port of xxxx and a source port of yyyy.

```
V TCPIP,tcpproc,PKTTRACE,DEST=xxxx,SRCP=yyyy
```

You can use the VARY TCPIP,tcpproc,OBEYFILE command to make temporary dynamic changes to system operation and network configuration without stopping and restarting the TCP/IP address space. For example, if you started the address space TCPIPA and created a sequential data set USER99.TCPIP.OBEYFIL1 containing packet trace statements, issue the following command:

```
VARY TCPIP,TCPIPA,CMD=OBEYFILE,DSN=USER99.TCPIP.OBEYFIL1
```

The VARY TCPIP,PKTTRACE command is cumulative. You can trace all packets for specified IP addresses by entering multiple PKTTRACE commands. In the following example, the two commands will trace all the packets received and all the packets sent for the specified IP addresses.

```
VARY TCPIP,,PKT,ON,IPADDR=(10.27.142.44)
VARY TCPIP,,PKT,ON,IPADDR=(10.27.142.45)
```

Formatting packet traces using IPCS

The IPCS CTRACE command parameters are described in “Formatting component traces” on page 58. The following notes apply to the IPCS CTRACE parameters with regard to the packet trace formatter:

JOBLIST, JOBNAME

The LINKNAME and JOBNAME keywords in the OPTIONS string can also be used to select records.

TALLY

Equivalent to the STATISTICS(DETAIL) option.

START and STOP

Packets are numbered after the START keyword has filtered records.

LIMIT

See the RECORDS keyword in the OPTIONS string.

USEREXIT

The packet trace formatter calls the CTRACE USEREXIT before testing the records with the filtering criteria. If it returns a nonzero return code, then the record will be skipped. The USEREXIT may also be used in the OPTIONS string. It is called after the record has met all the filtering criteria in the OPTIONS string.

COMP

Must be SYSTCPDA.

SUB

The SUB must name the TCP/IP procedure that created the CTRACE records when the input is a dump data set.

EXCEPTION

Since there are no EXCEPTION records for packet trace, the EXCEPTION keyword must not be specified.

ENTIDLIST

For packet trace, there can only be five values:

- 1 IPv4 packet trace records
- 2 X25 trace records
- 3 IPv4 Enterprise Extender data trace records
- 4 IPv6 packet trace records
- 5 All IPv4 and IPv6 data trace records except for IPv4 Enterprise Extender data trace

The CTRACE OPTIONS string provides a means of entering additional keywords for record selection and formatting packet traces (COMP=SYSTCPDA). See “Syntax” on page 59 for the complete syntax of CTRACE.

OPTIONS syntax

OPTIONS component

►►—OPTIONS—((—| Data Selection |—| Report Generation |—))—►►

Data Selection:

|—| Device Type |—| IP Identifier |—| IP Address |—| Name |—►
 ►|—| Port Number |—| Protocol |—| Record Number |—| Record Type |—|

Device Type:

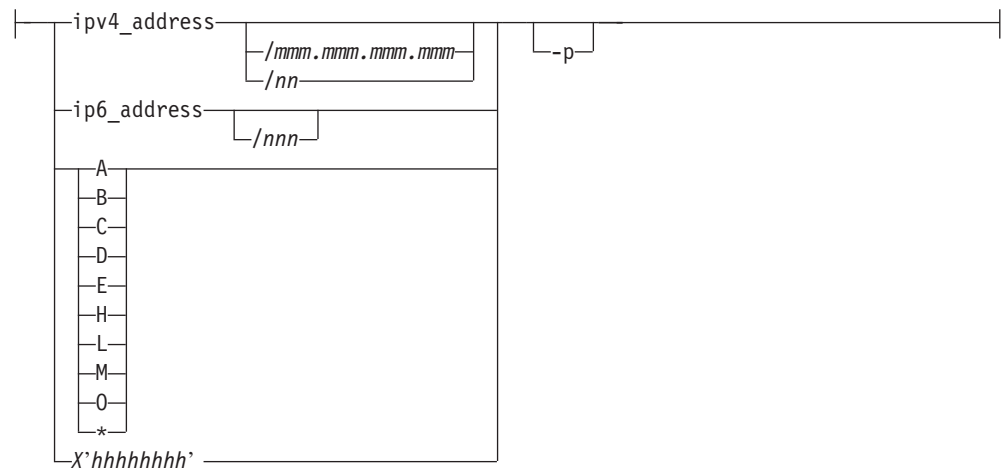
|—DEVTYPE—(—*devtype*—)—►

IP Identifier:

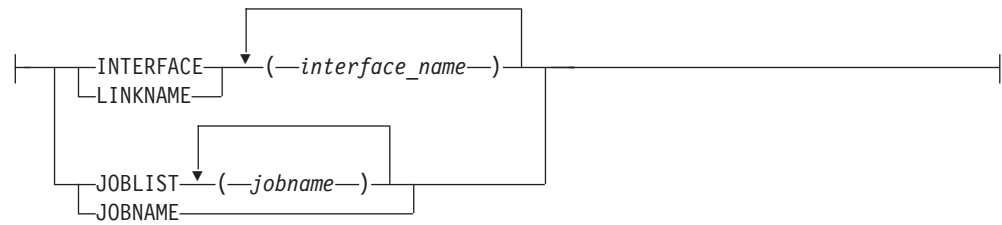
|—ADDR—(—| IP Address |—)—BROADCAST—CLASSA—CLASSB—CLASSC—CLASSD—►
 ►—CLASSE—HOST—IPADDR—(—| IP Address |—)—IPID—(—*ip_id_number*—)—►
 ►—IPV4—IPV6—LINKLOCAL—LOOPBACK—LOOPBACK6—MULTICAST—►

►—QOS—(—*quality_of_service*—)—SITELOCAL—TRAFFICCLASS—(—*traffic_class*—)—►

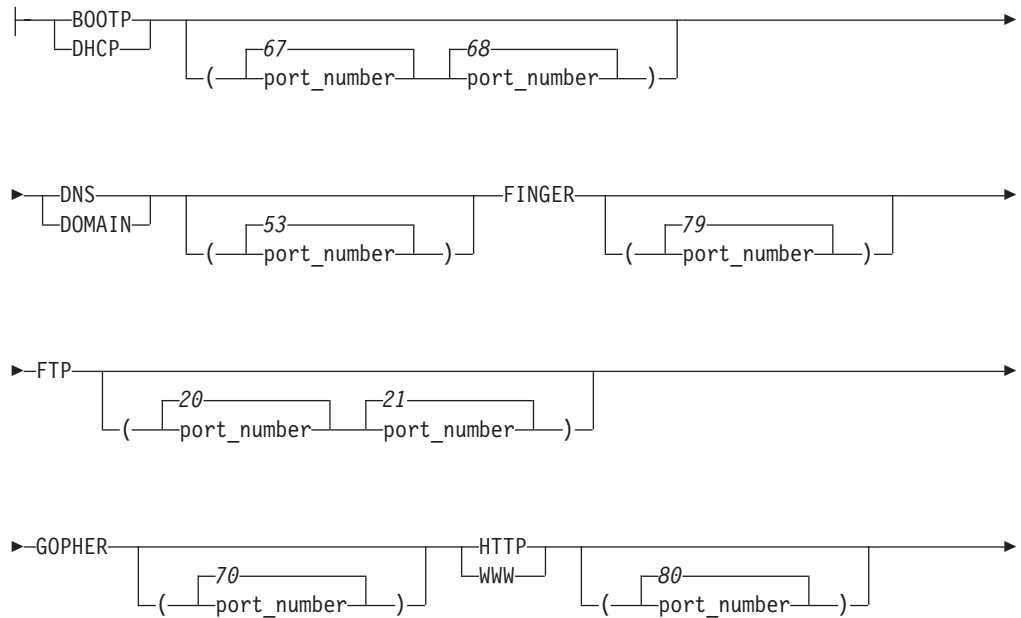
IP Address:

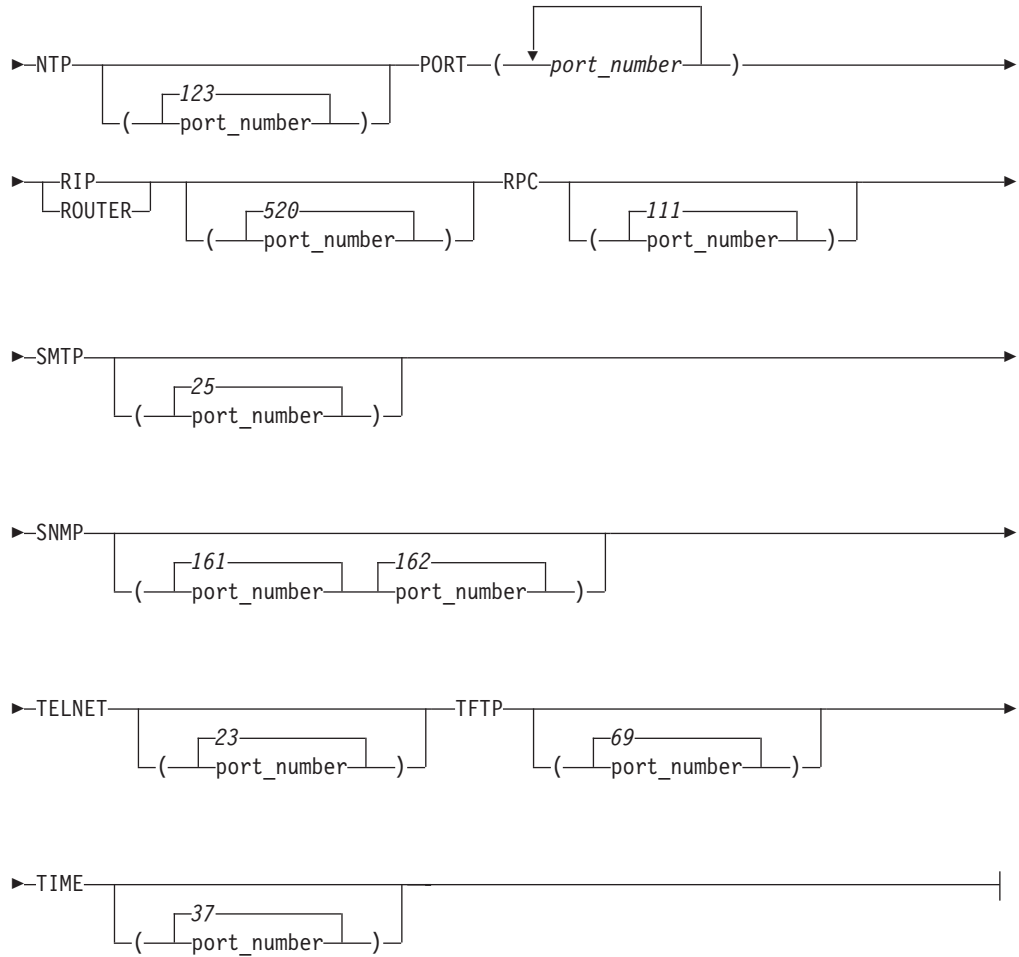
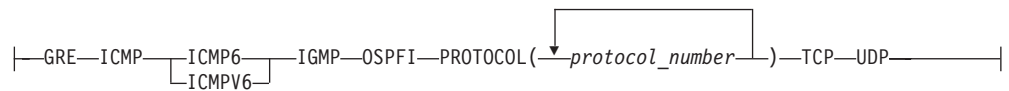
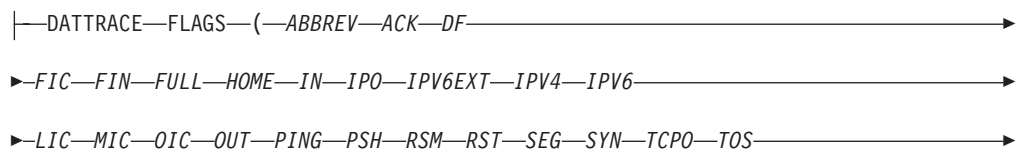


Name:



Port Number:



**Protocol:****Record Number:****Record Type:**

►TUNNEL—URG—ZWIN—) —PACKETTRACE—X25—USEREXIT(*exitname*) —
 —PKT—

Report Generation:

|
 —BOTH—
 —ASCII—
 —EBCDIC—
 —HEX—
 —BASIC—CLEANUP—
 —500—
 —(—nnnnn—)
 —DEBUG—(—nn—)

►—DELAYACK—(—200—
 —nnnn—)
 —DUMP—
 —(—65535—
 —nnnnn—)
 —EXPORT—
 —(—SUMMARY—
 —DETAIL—)

►—FMT—
 —FORMAT—
 —(—DETAIL—
 —SUMMARY—)
 —FULL—GAIN—(—125—
 —rtgain—, —150—
 —vargain—)

►—NOT—OPTION—
 —REASSEMBLY—
 —NOREASSEMBLY—
 —SEGMENT—
 —NOSEGMENT—

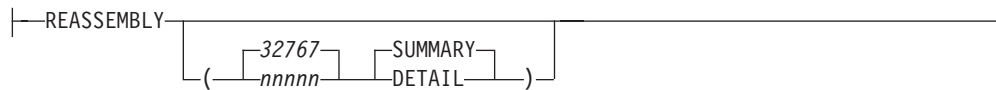
|
 ►—SESSION—
 —(—DETAIL—
 —SUMMARY—
 —STATE—)
 —SPEED—(—10—
 —local—, —10—
 —remote—)

►—SNIFFER—
 —(—200—
 —nnnnn—
 —TOKENRING—
 —ETHERNET—)

►—STATISTICS—
 —STATS—
 —(—SUMMARY—
 —DETAIL—)
 —TALLY—

|
 ►—STREAMS—
 —(—128—
 —nnnnn—
 —SUMMARY—
 —DETAIL—)
 —SUMMARY—TOD—

REASSEMBLY:



OPTIONS Keywords

The following are keywords used for the OPTIONS component routine parameters.

ASCII

Packet trace data dumped is shown in hexadecimal and interpreted in ASCII translation only. The default is BOTH.

BASIC

For specific packet types format each element of the packet data. Applies to DNS, RIP, and SNMP packet data.

BOOTP[(port_number|67 port_number|68)]

Select BOOTP and DHCP protocol packets. The port_number defines the BOOTP and DHCP port numbers to select packets for formatting. Equivalent to PORT(67 68).

BOTH

Packet trace data dumped is shown in hexadecimal and interpreted with both ASCII and EBCDIC translations. The default is BOTH.

BROADCAST

Select packets with a broadcast IPv4 address. Equivalent to IPADDR(255.255.255.255/255.255.255.255).

CLASSA

Select packets with a class A IPv4 address. Equivalent to IPADDR(0.0.0.0/128.0.0.0).

CLASSB

Select packets with a class B IPv4 address. Equivalent to IPADDR(128.0.0.0/192.0.0.0).

CLASSC

Select packets with a class C IPv4 address. Equivalent to IPADDR(192.0.0.0/224.0.0.0).

CLASSD

Select packets with a class D IPv4 address. Equivalent to IPADDR(224.0.0.0/240.0.0.0).

CLASSE

Select packets with a class E IPv4 address. Equivalent to IPADDR(240.0.0.0/248.0.0.0).

CLEANUP(nnnnn|500)

Defines a record interval where saved packet information in storage is released. The minimum value is 100 records; the maximum value is 1000000 records; the default is 500 records. If you set the record interval to 0, cleanup will not occur.

DEBUG(debug_level_list)

Provides documentation about SYSTCPDA format processing. debug_level_list is a list of numbers from 1 to 64. Use only under the direction of an IBM Service representative.

DELAYACK(threshold|200)

The delay acknowledgment threshold in milliseconds used in the calculation of

round trip time in the TCP session report. The minimum value is 10 milliseconds. The maximum value is 1000 milliseconds. The default value is 200 milliseconds.

DEVTYPE(device_type_list)

Select packets written to or received from an interface with one of the specified device types. From 1 to 16 types may be specified. This does not apply to data trace records. The following types can be specified:

- ATM
- CDLC
- CLAW
- CTC
- ETHER8023
- ETHERNET
- ETHEROR8023
- FDDI
- HCH
- IBMTR
- IPAQENET
- IPAQENET6
- IPAQIDIO
- IPAQTR
- LOOPBACK
- LOOPBACK6
- MPCPTP
- OSAFDDI
- OSAENET
- SNALINK
- SNALU62
- VIRTUAL
- VIRTUAL6
- X25NPSI

DNS[(port_number|53)]

Select Domain Namer Service protocol packets. The port_number defines the DNS port number to select packets for formatting. Equivalent to PORT(53).

DOMAIN[(port_number|53)]

Select Domain Namer Service protocol packets. The port_number defines the DNS port number to select packets for formatting. Equivalent to PORT(53).

DUMP[(nnnnn|65535)]

Dump the selected packets in hex with EBCDIC and ASCII translations. The IP and protocol headers are dumped separately from the packet data. The value *nnnnn* represents the maximum amount of packet data that is to be dumped from each packet. The default value is 65535 bytes. The minimum value is 0. The maximum value is 65535. The IP and protocol headers are not subject to this maximum.

The default report options are DUMP and FORMAT.

The BOTH, ASCII, EBCDIC and HEX keywords describe how the dumped packets are translated. The default is BOTH. The display can be changed using

these keywords. The default ASCII translation table is used. This table may not match the table being used by the application. When formatting the CTRACE, it is helpful to have the correct line length. Use the IPCS PROFILE LINESIZE command to set the line length. For example,

```
IPCS PROFILE LINESIZE(80)
```

sets the maximum line length to 80 characters so that all formatted data will be viewable within 80 characters.

If the STREAM report is chosen, then the dump of the packets is deferred until the stream of data has been collected.

EBCDIC

Packet trace data dumped is shown in hexadecimal and interpreted with EBCDIC translation only. The default is BOTH.

EXPORT[(DETAIL|SUMMARY)]

The selected packets are written to the EXPORT data set in .CSV (Comma Separated Value) format. In .CSV format, each character field is surrounded by double quotation marks and successive fields are separated by commas. The file's first line defines the fields. Each subsequent line is a record containing the values for each field.

DETAIL

Format the IP header, protocol header and protocol data as separate lines of data.

SUMMARY

Format the IP header and protocol header in one line of data. SUMMARY is the default.

Allocate a file with DDNAME of EXPORT before invoking the CTRACE command with EXPORT in the OPTIONS string.

```
ALLOC FILE(EXPORT) DA(PACKET.CSV) SPACE(15 15) TRACK
```

The record format is variable block with logical record length of 512 bytes.

FINGER[(port_number|79)]

Select FINGER protocol packets. The port_number defines the FINGER port number to select packets for formatting. Equivalent to PORT(79).

FLAGS(flags list)

Select packets that have the matching characteristics. Flags that can be specified are:

ABBREV

Select packets that are abbreviated.

ACK Select packets that have a TCP header with the ACK flag set.

DF Select IPv4 packets that have the don't fragment (ip_df) flag set.

FIC Select packets that are the first fragment of an IPv4 or IPv6 packet.

FIN Select packets that have a TCP header with the FIN flag set.

FULL Select packets that are complete.

HOME Select packets that have an IP destination address equal to the IP source address.

IN Select packets that are inbound.

IPO Select packets that have an IPv4 header options field.

IPV4 Select IPv4 packets. IPv4 cannot be used in combination with other data selectors that are IPv6-specific, such as LINKLOCAL.

IPV6 Select IPv6 packets. IPv6 cannot be used in combination with other data selectors that are IPv4-specific, such as BROADCAST.

IPV6EXT
Select packets that have an IPv6 extension header.

LIC Select packets that are the last fragment of an IPv4 or IPv6 packet.

MIC Select packets that are the middle fragment of an IPv4 or IPv6 packet.

OIC Select IPv4 or IPv6 packets that are not fragmented.

OUT Select packets that are outbound.

PING Select packets that are ICMP/ICMPv6 echo request and echo reply.

PSH Select packets that have a TCP header with the PSH flag set.

RSM Select packets that have been reassembled.

RST Select packets that have a TCP header with the RST flag set.

SEG Select packets that have been segmented.

SYN Select packets that have a TCP header with the SYN flag set.

TCPO Select packets that have a TCP header options field.

TOS Select IPv4 packets that have a nonzero value in the ip_tos field.

TUNNEL
Select packets with protocol number 47 GRE or 41 (IPv6 over IPv4).
z/OS Communications Server currently does not support IPv6 over IPv4 (protocol number 41).

URG Select packets that have a TCP header with the URG flag set.

ZWIN Select packets that have a TCP header with a zero window value.

Notes:

1. The use of the FIC, MIC and LIC flags require the use of the NOREASSEMBLY option.
2. When a packet is reassembled, then it becomes an OIC packet with the RSM flag set.

FMT

Equivalent to FORMAT.

FORMAT[(DETAIL|SUMMARY)]

The selected packets with defined packet data are to be formatted. The SHORT keyword on the CTRACE command selects this option if no other report options are specified. The default report options are DUMP and FORMAT.

DETAIL

Format the IP header, protocol header, and the protocol data.

SUMMARY

Format the IP header and protocol header. DETAIL is the default.

FTP[(data_port_number|20 control_port_number|21)]

Select FTP protocol packets. The port_number defines the FTP port numbers to select packets for formatting. Equivalent to PORT(20,21).

FULL

Equivalent to DUMP and FORMAT. The FULL keyword on the CTRACE command selects this option if no other report options are specified.

GAIN(rtgain|125,vargain|250)

Values of the round trip gain (rtgain) and the variance gain (vargain), in milliseconds, used in the calculation of round trip time in the TCP session report. Valid values are in the range 0–1000. The default values for rtgain is 125. The default value for vargain is 250.

GOPHER[(port_number|70)]

Select GOPHER protocol packets. The port_number defines the GOPHER port numbers to select packets for formatting. Equivalent to PORT(70).

GRE

Select packets with a protocol number of 47. Equivalent to PROTOCOL(47).

HEX

Packet trace data dumped is shown in hexadecimal only with no translation. The default is BOTH.

HOST

Select packets with a host IP address. Equivalent to IPADDR(0.0.0.0/255.255.0.0)

HTTP[(port_number|80)]

Select HTTP protocol packets. The port_number defines the HTTP port numbers to select packets for formatting. Equivalent to PORT(80). See the WWW keyword.

ICMP

Select packets with a protocol number of 1. Equivalent to PROTOCOL(1).

ICMP6 or ICMPV6

Select packets with a protocol number of 58. Equivalent to PROTOCOL(58).

IGMP

Select packets with a protocol number of 2. Equivalent to PROTOCOL(2).

INTERFACE(interface_name_list) or LINKNAME(interface_name_list)

Select packet trace records with the specified interface name. Up to 16 interface names can be specified. Each interface name can be up to 16 characters. An asterisk (*) may be used as a wild card to replace characters at the end of the interface name.

IPADDR(ipaddr[/mask_or_prefixlength][X'hhhhhhh'[]-nnnnn])

Select packets with a matching IP address, optional IPv4 address mask or IPv6 prefix length and optional port number. Up to 16 IP addresses can be specified. The IPADDR is specified in three parts:

1. An IPv4 or IPv6 address

The IPv4 address can be in dotted decimal notation, a keyword, or a hex value.

- IPv4 dotted decimal notation

127.0.0.1

- IPv4 keyword

A A class A IPv4 address, 0.0.0.0/128.0.0.0

B A class B IPv4 address, 128.0.0.0/192.0.0.0

C A class C IPv4 address, 192.0.0.0/224.0.0.0

- | | |
|----------|--|
| D | A class D IPv4 address, 224.0.0.0/240.0.0.0 |
| E | A class E IPv4 address, 240.0.0.0/248.0.0.0 |
| H | A local host address, 0.0.0.0/0.0.255.255 |
| L | An IPv4 or IPv6 loopback address, 127.0.0.0/255.0.0.0 or ::1 |
| M | The broadcast IPv4 address, 255.255.255.255/255.255.255.255 |
| * | Any address, 0.0.0.0/0.0.0.0 |
| 0 | An IPv4 or IPv6 address of zero, 0.0.0.0/255.255.255.255 or ::/128 |
- IPv4 or IPv6 address as a hexadecimal number up to 32 (IPv4) or 128 (IPv6) digits
X'7f000001'
 - IPv6 address
1080::8:800:200C:417A
2. An IPv4 address mask or IPv6 prefix length
The IPv4 address mask (1–32) or IPv6 prefix length (1–128) is preceded by a slash(/). Specify an IPv4 address mask only when the IPv4 address is in dotted decimal notation. The IPv4 address mask may be in dotted decimal notation, for example: 9.37/255.0.0.0 or 9.37/255.255.0.0
 3. A port number
The port number is preceded by a dash (-). It is a decimal number in the range 0–65535.

Notes:

1. There should be no spaces between the IP addresses and the subnet masks.
2. The BROADCAST, CLASSA, CLASSB, CLASSC, CLASSD, CLASSE, HOST, LINKLOCAL, LOOPBACK, MULTICAST, and SITELOCAL keywords add to the total of 16 IP addresses.
3. The port number when used adds to the the total of 16 port numbers in the PORT keyword.
4. IPv4 addresses and IPv4-mapped IPv6 addresses are treated as equivalent addresses.

IPID(ipid_number_list)

Select packets that match the ip_id number in the IPv4 packet header. Up to 16 ID numbers can be specified in the range 0–2147483647 or 0–X'FFFFFFF'. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnnnn) or hexadecimal (X'hhhhh'). If the packets have been fragmented, specify NOREASSEMBLY to select each packet.

IPv4

Equivalent to FLAGS(IPV4).

IPv6

Equivalent to FLAGS(IPV6).

JOBLIST|JOBNAME(job_name_list)

Select data trace records with the specified JOBNAME. Up to 16 job names can be specified. Each job name can be up to 8 characters. If the last character of a job name is an asterisk (*) then only the characters up to the asterisk are compared.

The CTRACE JOBLIST/JOBNAME parameter provides the same function, except that wildcards are not supported.

LINKLOCAL

Select packets with an IPv6 link-local unicast prefix. Equivalent to IPADDR(FE80::/10).

LINKNAME(link_name_list)

Select packet trace records with the specified LINKNAME. Up to 16 link names can be specified. Each link name can be up to 16 characters. If the last character of a link name is an asterisk (*) then only the characters up to the asterisk are compared.

The CTRACE JOBLIST/JOBNAME parameter provides the same function, except that wildcards are not supported and only the first 8 characters of the link name are compared.

LOOPBACK

Select packets with either an IPv4 or IPv6 loop back address. Equivalent to IPADDR(127.0.0.0/255.0.0.0::1). If other addresses are defined as loopback, they can be selected explicitly using IPADDR().

LOOPBACK6

Select packets with an IPv6 loop back address. Equivalent to IPADDR(::1).

MULTICAST

Select packets with either an IPv4 or IPv6 multicast address. Equivalent to CLASSD IPADDR(FF00::/8).

NOREASSEMBLY

Do not reassemble fragmented IP packets into a complete packet. REASSEMBLY is the default.

NOSEGMENT

Packet trace records that span multiple Ctrace records are not recombined. Only the first segment record of packet is used. The rest of the segment records are discarded. SEGMENT is the default.

NOT

If the NOT option is selected then any selection criteria is reversed. If a record matches the selection criteria, it is not processed. If a record does not match the selection criteria, it is processed.

NTP[(port_number|123)]

Select NTP protocol packets. The port number defines the NTP port number to select packets for formatting. Equivalent to PORT(123).

OPTION

The selected options with defaults are listed.

OSPF

Select packets with a protocol number of 89. Equivalent to PROTOCOL(89).

PACKETTRACE

Select packets written from the VARY TCPIP,,PKTTRACE command.

PKTTRACE

Equivalent to PACKETTRACE.

PORT(port_number_list)

Select packets with one of the specified port numbers. Up to 16 port numbers

can be specified in the range 0–65535. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnnnn) or hexadecimal (X'hhhh').

Note: The BOOTP, DHCP, DNS, DOMAIN, FINGER, GOPHER, HTTP, RIP, NTP, ROUTER, RPC, SMTP, SNMP, TELNET, TFTP, TIME and WWW keywords add to the list of 16 port numbers.

PROTOCOL(protocol number list)

Select packets with one of the specified protocol numbers. Up to 16 protocol numbers can be specified in the range 0–255. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnn) or hexadecimal (X'hh').

Protocol will filter on only the upper-layer header of an IPv6 packet. It will not filter for IPv6 extension headers (Hop-by-Hop Options, Routing, Fragment). Instead IPv6 extension headers will be included in the display of the basic IPv6 header.

Note: The GRE, ICMP, ICMP6, ICMPV6, IGMP, OSPFI, TCP and UDP keywords add to the list of 16 protocol numbers.

QOS(quality_of_service_list)

Select the records with the matching Quality of Service from the IPv4 Type of Service field. Up to 16 QoS values can be specified in the range 0–7. Each entry in the list can be a range: low_number:high_number. Values can be decimal (n) or hexadecimal (X'h').

REASSEMBLY[(packet_size|65535,DETAIL|SUMMARY)]

Reassemble IP fragments into a complete packet.

packet_size

The maximum size of a reassembled packet that is allowed. The smallest value allowed is 576 bytes, the largest is 65535 bytes. The default value is 65535 bytes.

DETAIL

List each of the reassembly statistics for each packet when a packet completes reassembly.

SUMMARY

Show only the reassembly statistics and information about packets that did not complete reassembly.

REASSEMBLY(65535,SUMMARY) is the default.

RECORDS(record_number_list)

Select the records with matching record numbers in the trace data. Up to sixteen (16) record numbers can be specified. Record numbers are assigned after any IPCS CTRACE selection criteria have been met. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnnnnnnnnn) or hexadecimal (X'hhhhhhhh').

RIP[(port_number|520)]

Select RIP protocol packets. The port_number defines the RIP port number to select packets for formatting. Equivalent to PORT(520).

ROUTER[(port_number|520)]

Select RIP protocol packets. The port_number defines the RIP port number to select packets for formatting. Equivalent to PORT(520).

RPC[(port_number|111)]

Select RPC protocol packets. The port_number defines the RPC port number to select packets for formatting. Equivalent to PORT(111).

SEGMENT

Packet trace records that span multiple Ctrace records are recombined. Data from segment records is saved until all the Ctrace records have been read to recreate the original packet. SEGMENT is the default.

SESSION[(DETAIL|STATE|SUMMARY)]

Generate a report showing TCP or UDP session traffic.

DETAIL

List each of the packets for a session, as well as the summary statistics. DETAIL is the default.

STATE

List the beginning and ending state of each session.

SUMMARY

Show only the summary statistics.

Note: The UDP session analysis is also used for other protocols.

SITELOCAL

Select packets with an IPv6 site-local unicast address prefix. Equivalent to IPADDR(FEC0::/10).

SMTP[(port_number|25)]

Select SMTP protocol packets. The port_number defines the SMTP port number to select packets for formatting. Equivalent to PORT(25).

SNIFFER[(TOKENRING|ETHERNET)]

Formats the trace records so that they are suitable for downloading to a Network Associates' Sniffer** Network Analyzer for analysis. The trace records are written to the file with a DD name of SNIFFER. Once the file is generated, it can be downloaded as a binary file to the analyzer and loaded using the standard features of the analyzer. Use NOREASSEMBLY to prevent the formatter from reassembling packets. Each packet is passed as the packets are collected. The logical record length of the SNIFFER data set determines the largest amount of packet data written to the data set. Allocate a file with DDNAME of SNIFFER before invoking the CTRACE command with SNIFFER in the OPTIONS string.

```
ALLOC FILE(SNIFFER) DA(PACKET.TRC) SPACE(15 15) TRACK +
      LRECL(8000) BLKSIZE(32000)
```

The data set has a record format of variable blocked with a logical record length of 8000 bytes. The maximum IP packet size is 7954 (8000 - 46) for SNIFFER(TOKENRING) and the maximum P packet size is 7962 (8000 - 38) for SNIFFER(ETHERNET).

TOKENRING

If this keyword is specified, the output is formatted for the token-ring analysis application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on a token ring is collected. Packets from all devices can be collected using this option. If you are using the token-ring analyzer application, the DOS file extension must be TRC.

ETHERNET

If this keyword is specified, the output is formatted for the Ethernet analysis

application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on an Ethernet are collected. Packets from all devices can be collected using this option. If you are using the Ethernet analyzer application, the DOS file extension must be ENC.

The default for the SNIFFER option is TOKENRING.

SNMP[(port_number|161 port_number|162)]

Select SNMP protocol packets. The port_number defines the SNMP port number to select packets for formatting. Equivalent to PORT(161 162).

SPEED(local|10,remote|10)

The link speed, in megabits per second, for the local and remote link. These values are used in throughput calculations in the TCP session report. Valid values are in the range 0–18010. The default value is 10. Specify the slowest speed of the link in the route.

STATISTICS[(DETAIL|SUMMARY)]

After all the records have been processed, generate statistical reports.

DETAIL

Reports are produced showing the number of records selected by record type, device type, jobname, linkname, protocol number, IP address and port numbers. The session summary report is a listing of the IP address and port number pairs showing the number of records, the first and last record numbers, and the first and last record times.

SUMMARY

Only the session summary report is produced. SUMMARY is the default.

TALLY on the CTRACE command selects this option if no other report options are specified.

STATS

Equivalent to the STATISTICS option.

STREAMS[(stream_size|128 DETAIL|SUMMARY)]

Collect the packet data for dumping or formatting after the trace file is processed. The value *nnn* represents the maximum amount of storage used to capture each stream. The value *stream_size* represents the maximum amount of storage used to capture each stream. The smallest value is 16KB. The largest value is 512KB. The default value is 128KB. The value is in 1024 bytes (1K) units.

SUMMARY

List about each packet in the stream. SUMMARY is the default.

DETAIL

Issue messages about the status of the stream.

Note: The DUMP keyword is required to dump the packet data.

SUMMARY

Format a single line for each trace record. SUMMARY on the CTRACE command selects this option if no other report options are specified. If no other report option specified on the CTRACE command, then SUMMARY is selected as the report.

TALLY

Equivalent to the STATISTICS(DETAIL) option.

TCP

Select packets with a protocol number of 6. Equivalent to `PROTOCOL(6)`.

TELNET[(port_number|23)]

Select TELNET protocol packets. The `port_number` defines the TELNET port number to select packets for formatting. Equivalent to `PORT(23)`.

TFTP[(port_number|69)]

Select TFTP protocol packets. The `port_number` defines the TFTP port number to select packets for formatting. Equivalent to `PORT(69)`.

TIME[(port_number|37)]

Select TIME protocol packets. The `port_number` defines the TIME port number to select packets for formatting. Equivalent to `PORT(37)`.

TOD

Use the time the trace data was captured for the reports. Normally the time the trace data was moved to the trace buffer is shown.

Note: The CTRACE command uses the time stamp when the trace data was moved to the buffers for START and STOP time selection.

TRAFFICCLASS(traffic_class)

Select the records with the matching IPv6 traffic class field. Up to 16 traffic class values can be specified in the range from 0 to 255. Each entry in the list can be a range: `low_number:high_number`. Values can be decimal (nn) or Hexadecimal (X'hh').

UDP

Select packets with a protocol number of 17. Equivalent to `PROTOCOL(17)`.

USEREXIT(exitname)

Names the user exit to be called for each selected record. The USEREXIT keyword on the CTRACE command names a user exit that is called before the SYSTCPDA packet trace filtering is done. If this exit routine returns a nonzero return code, then the record is skipped by the SYSTCPDA formatter.

WWW[(port_number|80)]

Select HTTP protocol packets. The `port_number` defines the HTTP port number to select packets for formatting. Equivalent to `PORT(80)`.

X25

Select packet trace records created by the X25 processor.

Report Examples

The CTRACE packet trace (SYSTCPDA) report generation outputs are described in the following examples.

Because IPv6 increases the IP address size, formatted IPv6 packet/data traces might be much wider than 80 columns.

OPTION:

Purpose: List the selected options and default keyword values.

Format: CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((STAT(DETAIL) OPTION TCP))

Examples:

```
COMPONENT TRACE SHORT FORMAT
COMP(SYSTCPDA)SUBNAME((TCPCS))
OPTIONS((STAT(DETAIL) OPTION TCP))
z/OS TCP/IP Packet Trace Formatter, (C) IBM 2000-2002, 2001.352
1 DSNAME('IPCS.R744334.DUMPA')

2 OPTIONS((Both Bootp(67,68) Cleanup(500) DelayAck(200,200) Domain(53)
Finger(79) Flags() Ftp(20,21) Gain(125,250) Gopher(70) Ntp(123) Option
Reassembly(65535,Summary) Router(520) Rpc(111) Segment Sntp(25)
Snmp(161,162) Speed(10,10) Statistics(Detail) Telnet(23) Tftp(69) Time(37)
Userexit() Www(80)
3 Protocol( /* 1 */
6 /* TCP */,
)
))
```

The following fields are on the OPTION report.

- 1 DSNAME - The name of the source data.
- 2 OPTIONS((...)) - A listing of the active options with default values.
- 3 When a filter is specified, the list of filters with the number of filter values and filter values.

SUMMARY:

Purpose: Show one or two lines of information about each record in the trace.

Format: CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SUMMARY

Examples: The following fields are on the SUMMARY report.

```
**** 2002/02/12
I - Inbound packet
O - Outbound packet

DP  Nr hh:mm:ss.mmmmmm IpId   Seq_num   Ack_num   Wndw  Flags           DatLn Data           Destination/Source
IU   4 19:49:48.768707 0068                                43 30290201 9.67.115.5-1035
II   5 19:49:48.768830 0069 ICMP_UNREACH_PORT=161          28          9.67.115.5-161
IU   6 19:49:51.769220 006A                                43 30290201 9.67.115.5-1035
II   7 19:49:51.769350 006B ICMP_UNREACH_PORT=161          28          9.67.115.5-161
IU   8 19:49:57.769780 006C                                43 30290201 9.67.115.5-1035
II   9 19:49:57.769909 006D ICMP_UNREACH_PORT=161          28          9.67.115.5-161
IU  10 19:51:06.775919 006E                                43 30290201 9.67.115.5-1036
OU  40 19:52:39.284802 0000                                12 01010000 FEC9:C2D4::6:2900:EDC:217C-32810
OI  41 19:52:39.284870 0000 ICMPv6_Neighbor Solicitation  24          FEC9:C2D4::9:67:115:17-33435
II  42 19:52:39.285955 0000 ICMPv6_Neighbor Advertisement  24          FEC9:C2D4::9:67:115:17
II  43 19:52:39.302034 0000 ICMPv6_Port unreachable       60          FEC9:C2D4::6:2900:EDC:217C
OU  44 19:52:39.310642 0000                                12 02010000 FEC9:C2D4::6:2900:EDC:217C-32810
                                         FEC9:C2D4::9:67:115:17-33436
```

D Direction of the packet:

I Inbound

O Outbound

P The packet protocol

T TCP

U UDP

I ICMP

G IGMP

D Data Trace

P Neither TCP, UDP, ICMP, nor IGMP

Nr The Ctrace record number

hh:mm:ss.mmmmmm

The time stamp of the record

Source

The source IP address and port number

Destination

The destination IP address and port number

IpId

The packet ID number in hexadecimal

- For TCP

- seq_num**
The sequence number
- ack_num**
The acknowledgment sequence number
- wndw**
The window size
- flags**
The TCP header flags
- DatL**
The length of data in the datagram
- EBCDIC**
The first eight bytes with EBCDIC translation
- ASCII**
The first eight bytes with ASCII translation
- For UDP
 - nnnnn**
The length of the UDP datagram
 - DatL**
The length of the UDP packet data
 - EBCDIC**
The first eight bytes with EBCDIC translation
 - ASCII**
The first eight bytes with ASCII translation
- For ICMP
 - cccccccccc**
The type of ICMP message
 - xxxxxxxxxx**
The first eight bytes of the user data in hex
- For IGMP
 - nnnnn**
The maximum response time
 - cccccccccc**
The type of IGMP message
 - nnn.nnn.nnn.nnn**
The IGMP group address
- Other protocols
 - cccccccccc**
The protocol name
 - nnnnn**
The length of the protocol data

EXPORT:

Purpose: Reformat the information about the IP header, protocol header, and packet data into a file with CSV format.

Format:

```
ALLOC FILE(EXPORT) DA(EXPORT.CSV) SPACE(15 15) TRACK
CTTRACE COMP(SYSTCPDA) SUB((TCPDS)) SHORT OPTIONS((EXPORT))
```

Examples: The following describe the EXPORT, EXPORT(SUMMARY), and EXPORT(DETAIL) report outputs.

• EXPORT

Export Report

- 1** 124 records written to USER2.EXPORT.CSV
- 2** 20,168 bytes written

The following fields are on the EXPORT report.

1

The number of data records written to the export data set.

2

The size of the export data set.

• EXPORT (SUMMARY)

```
"Flags ","Packet","Absolute Time ","Rel Time","Delta Time",
"Device ","Source ","Destination ","
"Id","IpLen","Protocol ","Summary"
"I 0 ", 1,"19:49:42.788207", 0.000000, 0.000000,
"OSAQDIOLINK ", "9.67.115.17 ", "9.67.115.63 ",
17158, 78, "UDP", "S=137 D=137 LEN=58"
"I 0 ", 29, "19:52:21.240160", 158.451952, 0.016739,
"OSAQDIOLINK ", "9.67.115.69 ", "9.67.115.5 ",
5971, 56, "ICMP", "? LEN=28"
"I 0 ", 37, "19:52:27.783944", 164.995736, 0.000134,
"LOOPBACK ", "9.67.115.5 ", "9.67.115.5 ",
129, 56, "ICMP", "? LEN=28"
"O 0 ", 40, "19:52:39.284802", 176.496595, 5.500260,
"OSAQDI046 ", "FEC9:C2D4::6:2900:EDC:217C", "FEC9:C2D4::9:67:115:17",
20, 60, "UDP", "S=32810 D=33435 LEN=20"
"O 0 ", 41, "19:52:39.284870", 176.496662, 0.000067,
"OSAQDI046 ", "FEC9:C2D4::6:2900:EDC:217C", "FF02::1:FF15:17",
32, 72, "ICMPv6", "ICMPv6"
"I 0 ", 42, "19:52:39.285955", 176.497748, 0.001085,
"OSAQDI046 ", "FEC9:C2D4::9:67:115:17", "FEC9:C2D4::6:2900:EDC:217C",
32, 72, "ICMPv6", "ICMPv6"
"O 0 ", 49, "19:52:58.286347", 195.498140, 13.972912,
"LOOPBACK6 ", "FEC9:C2D4::9:67:115:5", "FEC9:C2D4::9:67:115:5",
20, 60, "UDP", "S=32810 D=33435 LEN=20"
"I 0 ", 50, "19:52:58.286530", 195.498323, 0.000182,
"LOOPBACK6 ", "FEC9:C2D4::9:67:115:5", "FEC9:C2D4::9:67:115:5",
68, 108, "ICMPv6", "ICMPv6"
```

The following describes fields found on the EXPORT (SUMMARY) report:

Control flags

Direction

- I — Input
- O — Output

A The packet was abbreviated (used with the following fragment flags).

- R** Reassembled packet.
- O** The Only fragment of a packet (it is complete).
- F** First fragment of a packet.
- M** Middle fragment of a packet.
- L** Last fragment of a packet.

T The packet was in a tunnel.

Packet

The packet number

Absolute Time

The time stamp on the packet

Rel Time

The time from the first packet in seconds

Delta Time

The time from the previous packet in seconds

Device

The device the packet was received on or sent from

Source

The source IP address

Destination

The destination IP address

IpId

The ID number from the IP packet header

IpLen

The length of the IP packet

Protocol

The protocol from the IP packet

Summary

Additional information from the protocol header.

• EXPORT (DETAIL)

```
"Flags ","Packet","Delta Time","Source ","
"Destination ","Protocol ","Summary"
"I O ", 10," 69.006010","9.67.115.5 ",
"9.67.115.5 ", "IP", " S=9.67.115.5 D=9.67.115.5 LEN=71 ID=110"
"I O ", 10," 69.006010","9.67.115.5 ",
"9.67.115.5 ", "UDP", " S=1036 D=161 LEN=51"
"I O ", 10," 69.006010","9.67.115.5 ",
"9.67.115.5 ", "SNMP", "GetRequest dpiPathNameForUnixStream.0"
"O O ", 24," 0.002956","9.67.115.5 ",
"9.67.115.69 ", "IP", " S=9.67.115.5 D=9.67.115.69 LEN=40 ID=121"
"O O ", 24," 0.002956","9.67.115.5 ",
"9.67.115.69 ", "UDP", " S=32810 D=33436 LEN=20"
"O O ", 51," 0.002695","FEC9:C2D4::9:67:115:5",
"FEC9:C2D4::9:67:115:5", "IP", " S=FEC9:C2D4::9:67:115:5 D=FEC9:C2D4::9:67:115:5 LEN=60 ID=20"
"O O ", 51," 0.002695","FEC9:C2D4::9:67:115:5",
"FEC9:C2D4::9:67:115:5", "UDP", " S=32810 D=33436 LEN=20"
"I O ", 52," 0.000244","FEC9:C2D4::9:67:115:5",
"FEC9:C2D4::9:67:115:5", "IP", " S=FEC9:C2D4::9:67:115:5 D=FEC9:C2D4::9:67:115:5 LEN=108 ID=68"
"I O ", 52," 0.000244","FEC9:C2D4::9:67:115:5",
"FEC9:C2D4::9:67:115:5", "ICMPv6"
```

The following describes fields found on the EXPORT (DETAIL) report:

Control flags

Direction

- I — Input
- O — Output

A The packet was abbreviated (used with the following fragment flags).

R Reassembled packet.

O Only fragment of a packet (it is complete).

F First fragment of a packet.

M Middle fragment of a packet.

L Last fragment of a packet.

T The packet was in a tunnel.

Packet

The packet number.

Delta Time

The time from the previous packet in seconds.

Source

The source IP address.

Destination

The destination IP address.

Protocol

There are multiple lines about a single packet. The first line contains "IP" to identify the data in the summary field. The second line identifies information about the protocol used by the packet. The possible third line identifies the application data in the packet.

Summary

Additional information from the protocol headers or packet data.

FORMAT:

Purpose: Format the Ctrace record header, the IP packet header, the protocol header, and the packet data. If one of the ports is a well-known port number and the SYSTCPDA supports data for the port number, the packet data is shown.

Format:

```
CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((FORMAT))
```

Examples:

```
1 3 MVSJ      PACKET  00000001 23:39:11.873541 Packet Trace
2 To Interface : TR1      Device: LCS Token Ring   Full=56
  Tod Clock    : 2002/02/12 23:39:11.873539
  Sequence #   : 0        Flags: Pkt Ver2 Out
  Source Port  : 1025     Dest Port: 53   Asid: 001E TCB: 007F62C0
3 IpHeader: Version : 4      Header Length: 20
  Tos          : 00        QOS: Routine Normal Service
  Packet Length : 56      ID Number: 000E
  Fragment     :          Offset: 0
4 TTL          : 64        Protocol: UDP      CheckSum: A6FB FFFF
  Source       : 9.67.113.65
  Destination  : 9.37.80.3

5 UDP
  Source Port   : 1025  ()      Destination Port: 53   (domain)
  Datagram Length : 36      CheckSum: AD0B FFFF
6 DNS: 28

=====
7 ;; -->DNS HEADER<<- opcode: QUERY, status: NOERROR, id: 40266
;; flags: rd; Ques: 1, Ans: 0, Auth: 0, Addit: 0

;; QUESTIONS: 1
;; w3.ibm.com          IN      AAAA
```

1

A summary line indicating the source of the trace record showing:

- The record number.
- The system name.
- The type of the trace record.
- The time the record was moved to the trace buffer, or with the TOD option the time the trace data was captured.
- The description of the trace record, Packet Trace, X25, or Data Trace.

2

The trace header with these fields:

- The direction of the trace record: From or To.
- The network interface name (or job name for Data Trace).
- The device type.
- Full or Abbrev with amount of trace data available.
- The time the trace record was captured.
- The number of records lost.
- The packet trace header flags.

3

The IP header showing fields from the IP version 4 header. The header length is the number bytes for the header. The offset field is the number of bytes from

the end of the IP header of where the fragment appears. With the REASSEMBLY option active, this field always contains zeros.

4 The check sum value. If possible, the check sum of the packet is calculated. If the calculated value is X'FFFF', the check sum is correct. If X'0000', the check sum could not be calculated because the packet was incomplete or fragmented. Other values indicate a check sum error.

5 The UDP protocol header. The fields of the header are shown.

6 The length of the DNS packet data following is shown.

7 The DNS header and resource records are formatted. Using the protocol numbers and the well known port numbers, format routines are invoked to format standard packet data records. The port number for the PORT keywords define the port numbers to be used to invoke a format routine.

Port	Keyword
------	---------

67, 68	BOOTP
--------	-------

67, 68	DHCP
--------	------

53	Domain
----	--------

520	RIP
-----	-----

520	Router
-----	--------

161,162	SNMP
---------	------

23	TELNET
----	--------

69	TFTP
----	------

DUMP:

Purpose: Format the IP header, protocol header, and packet data in hex. The data can also be translated into EBCDIC, ASCII, or both.

Format:

```
CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((DUMP))
```

Examples:

```
1 MVS073  PACKET  00000001 19:49:42.788207 Packet Trace
  From Interface   : OSAQDIOLINK      Device: QDIO Ethernet    Full=78
  Tod Clock        : 2002/02/12 19:49:42.788204
  Sequence #       : 0                 Flags: Pkt Ver2
  Source Port      : 137                Dest Port: 137    Asid: 002B TCB: 00000000
```

```
1 IP Header           : 20
000000 4500004E 43060000 8011FEC2 09437311 0943733F
```

```
2 Protocol Header     : 8
000000 00890089 003AD7D7
```

```
3 Data                : 50      Data Length: 50
000000 AD3D0110 00010000 00000000 20464845 ..... FHE
000010 50464345 4C454846 43455046 46464143 &...<....&.... PFCELEHFCEPFFFAC
000020 41434143 41434143 41434142 4C000020 .....<... ACACACACACABL..
000030 0001                                .. ..
```

1

The IP header is dumped with no translation.

2

The protocol header is dumped with no translation.

3

The packet data is dumped with the translation specified by the ASCII, BOTH, EBCDIC or HEX keyword. The default is BOTH. The amount of data dumped can be limited by the value specified with the DUMP keyword. The default is 65535 bytes.

REASSEMBLY:

Purpose: This report shows the packets that were reassembled. Use the REASSEMBLY(DETAIL) option to see all the packets that were reassembled. If the reassembled packets are larger than 32K then use REASSEMBLY(nnnnn), where nnnnn is the maximum size of a reassembled packet.

Format:

```
CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((REASSEMBLY(DETAIL) STAT))
```

Examples:

```
1 Reassembly of: 9.67.113.65-0 9.27.11.173-0 Id: 0043 status: +Fic +Lic
2  Rcd Nr Time          Delta          Offset Length  Next    Gap  Data Flags
   1638 15:28:49.975479 00:00:00.000000      0    3976   3976    0  3976 Fic
   1639 15:28:49.975501 00:00:00.000022   3976   3976   7952    0  3976 Mic
   1640 15:28:49.975524 00:00:00.000044   7952   3976  11928    0  3976 Mic
   1641 15:28:49.975545 00:00:00.000066  11928   3976  15904    0  3976 Mic
   1642 15:28:49.975567 00:00:00.000088  15904   3976  19880    0  3976 Mic
   1643 15:28:49.975594 00:00:00.000115  19880   3976  23856    0  3976 Mic
   1644 15:28:49.975620 00:00:00.000141  23856   3976  27832    0  3976 Mic
   1645 15:28:49.975689 00:00:00.000210  27832   3976  31808    0  3976 Mic
   1646 15:28:49.975737 00:00:00.000258  31808   3976  35784    0  3976 Mic
   1647 15:28:49.975771 00:00:00.000292  35784   3976  39760    0  3976 Mic
   1648 15:28:49.975804 00:00:00.000325  39760   3976  43736    0  3976 Mic
   1649 15:28:49.975835 00:00:00.000356  43736   3976  47712    0  3976 Mic
   1650 15:28:49.975865 00:00:00.000386  47712   3976  51688    0  3976 Mic
   1651 15:28:49.975898 00:00:00.000419  51688   3976  55664    0  3976 Mic
   1652 15:28:49.975926 00:00:00.000447  55664   3976  59640    0  3976 Mic
   1653 15:28:49.975962 00:00:00.000483  59640   3976  63616    0  3976 Mic
   1654 15:28:49.975986 00:00:00.000507  63616    392  64008    0   392 Lic
64,008 bytes is the final length of the IP packet
17 packets were used for reassembly
64,008 bytes were accumulated for reassembly
=====
3 Packet Reassembly Report
Maximum reassembly buffer size is 65535
4 Reassembly of: 9.27.11.173-0 9.67.113.65-0 Id: 3694 status: +Fic +Lic
   Rcd Nr Time          Delta          Offset Length  Next    Gap  Data Flags
   1655 15:28:50.024685 00:00:00.000000      0   1480   1480    0  1480 Fic
   1656 15:28:50.024705 00:00:00.000019   1480   1480   2960    0  1480 Mic
   1657 15:28:50.024739 00:00:00.000053   2960   1480   4440    0  1480 Mic
   1658 15:28:50.024772 00:00:00.000086   4440   1480   5920    0  1480 Mic
   1659 15:28:50.025506 00:00:00.000820   5920   1480   7400    0  1480 Mic
   1660 15:28:50.030534 00:00:00.005848   7400   1480   8880    0  1480 Mic
   1661 15:28:50.030592 00:00:00.005906   8880   1480  10360    0  1480 Mic
   1662 15:28:50.030607 00:00:00.005921  10360   1480  11840    0  1480 Mic
   1663 15:28:50.030650 00:00:00.005964  11840   1480  13320    0  1480 Mic
   1664 15:28:50.030683 00:00:00.005997  13320   1480  14800    0  1480 Mic
   1665 15:28:50.030698 00:00:00.006012  14800   1480  16280    0  1480 Mic
   1666 15:28:50.042927 00:00:00.018241  16280   1480  17760    0  1480 Mic
   1667 15:28:50.042946 00:00:00.018261  17760   1480  19240    0  1480 Mic
   1668 15:28:50.043006 00:00:00.018320  19240   1480  20720    0  1480 Mic
   1669 15:28:50.043021 00:00:00.018335  20720   1480  22200    0  1480 Mic
   1670 15:28:50.043058 00:00:00.018372  22200   1480  23680    0  1480 Mic
   1671 15:28:50.043114 00:00:00.018428  23680   1480  25160    0  1480 Mic
   1672 15:28:50.043130 00:00:00.018444  25160   1480  26640    0  1480 Mic
   1673 15:28:50.043174 00:00:00.018488  26640   1480  28120    0  1480 Mic
   1674 15:28:50.043222 00:00:00.018536  28120   1480  29600    0  1480 Mic
   1675 15:28:50.043257 00:00:00.018571  29600   1480  31080    0  1480 Mic
   1676 15:28:50.043544 00:00:00.018858  31080   1480  32560    0  1480 Mic
   1677 15:28:50.043592 00:00:00.018906  32560   1480  34040    0  1480 Mic
   1678 15:28:50.043607 00:00:00.018921  34040   1480  35520    0  1480 Mic
   1679 15:28:50.044618 00:00:00.019932  35520   1480  37000    0  1480 Mic
   1680 15:28:50.044649 00:00:00.019963  37000   1480  38480    0  1480 Mic
   1681 15:28:50.044698 00:00:00.020012  38480   1480  39960    0  1480 Mic
```

```

1682 15:28:50.044712 00:00:00.020026 39960 1480 41440 0 1480 Mic
1683 15:28:50.044745 00:00:00.020059 41440 1480 42920 0 1480 Mic
1684 15:28:50.044778 00:00:00.020092 42920 1480 44400 0 1480 Mic
1685 15:28:50.050178 00:00:00.025492 44400 1480 45880 0 1480 Mic
1686 15:28:50.050212 00:00:00.025527 45880 1480 47360 0 1480 Mic
1687 15:28:50.050244 00:00:00.025558 48840 1480 50320 -1480 1480 Mic
1688 15:28:50.050275 00:00:00.025589 50320 1480 51800 0 1480 Mic
1689 15:28:50.050328 00:00:00.025642 51800 1480 53280 0 1480 Mic
1690 15:28:50.050343 00:00:00.025657 53280 1480 54760 0 1480 Mic
1691 15:28:50.054558 00:00:00.029872 54760 1480 56240 0 1480 Mic
1692 15:28:50.054614 00:00:00.029928 57720 1480 59200 -1480 1480 Mic
1693 15:28:50.054628 00:00:00.029942 59200 1480 60680 0 1480 Mic
1694 15:28:50.054680 00:00:00.029994 62160 1480 63640 -1480 1480 Mic
1695 15:28:50.054694 00:00:00.030008 63640 368 64008 0 368 Lic
64,008 bytes is the final length of the IP packet
41 packets were used for reassembly
59,568 bytes were accumulated for reassembly
-----

```

```

5 1,641 packets required reassembly
54 IP packet reassemblies were done
52 IP packets were completely reassembled
2 IP packets were incomplete
0 packets failed reassembly
1,627 storage requests for buffers were made
64,080 bytes of buffer space are still in use
191,872 bytes of buffer space was the maximum in use
114,688 bytes of control storage were used

```

Reassembly is always done (except with the NOREASSEMBLY option). However, the REASSEMBLY(DETAIL) option is needed for the report on completed reassemblies.

1

The current packet that was reassembled is identified with source and destination IP address and port numbers. The IP identification number is shown. The status of the reassembly is shown. Completed packets are shown when the final packet is received. Incomplete packets are shown during the final processing.

2

Each packet that was reassembled is shown. The flag shows type of packet:

Fic First in chain. The offset was zero.

Mic Middle in chain. The offset was nonzero and the more fragment flag was set.

Lic Last in chain. The offset was nonzero and the the more fragment flag was not set.

Ooo The packet arrived out of order.

The Gap field is the number of bytes between the end of one packet and the start of the next. This should have a value of zero for normal processing. Nonzero values indicate duplicate data being sent.

3

When all the trace records have been processed the final report on reassembly is formatted. The maximum reassembly buffer size is shown. Packets that would exceed this size are rejected. This simulates the Ping of Death processing.

4

Incomplete packets that did not complete reassembly are shown.

The total number of trace records that were reassembled is shown with other statistics.

200 packets required reassembly

The number of packets that required reassembly (that had a fragment offset or the more fragment flag set).

57 IP packet reassemblies were done

The number of reassembled packets.

54 IP packets were completely reassembled

The number of reassembled packets where all the fragments were found.

3 IP packets were incomplete

The number of reassembled packets where all the fragments were not found.

0 packets failed reassembly

The number of packets that would have caused the completed packet to exceed the reassembly size.

170 storage requests for buffers were made

The number of times a request for reassembly buffer was made.

128,747 bytes of buffer space is still in use

The amount of storage still in use for incomplete packets.

284,158 bytes of buffer space was the maximum in use

The maximum amount of storage in use while reassembling packets.

Note: For reassembled packets the calculated check sum fields are not X'FFFF', because the packets were modified by the reassembly process.

SESSION:

Purpose: This report shows traffic for a TCP session.

Format:

CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((SESSION TCP))

Examples:

```
-----
1 2 packets summarized
Local Ip Address:      FEC9:C2D4::6:2900:EDC:217C
Remote Ip Address:     FEC9:C2D4::9:67:115:17
Host:                  Local,          Remote
Client or Server:      CLIENT,         SERVER
Port:                  1027,           21
Application:            ,              ftp
Link speed (parm):     10,             10 Megabits/s
2 Connection:
First timestamp:       19:55:46.934032
Last timestamp:        19:55:46.934989
Duration:              00:00:00.000957
Average Round-Trip-Time: 0.000 sec
Final Round-Trip-Time: 0.000 sec
Final state:           CLOSED (PASSIVE RESET)
Out-of-order timestamps: 0
3 Data Quantity & Throughput:
Inbound,              Outbound
Application data bytes: 0,              0
Sequence number delta: 0,              1
Total bytes Sent:      0,              0
Bytes retransmitted:    0,              0
Throughput:            0,              0 Kilobytes/s
Bandwidth utilization: 0.00%,          0.00%
Delay ACK Threshold:   200,            200 ms
Minimum Ack Time:      0.000957,        0.000000
Average Ack Time:      0.000957,        0.000000
Maximum Ack Time:      0.000957,        0.000000
4 Data Segment Stats:
Inbound,              Outbound
Number of data segments: 0,              0
Maximum segment size:   1432,            0
Largest segment size:   0,              0
Average segment size:   0,              0
Smallest segment size:  0,              0
Segments/window:        0.0,            0.0
Average bytes/window:   0,              0
Most bytes/window:      0,              0
5 Window Stats:
Inbound,              Outbound
Number of windows:      0,              0
Maximum window size:    0,              0
Largest window advertised: 0,            32768
Average window advertised: 0,            32768
Smallest window advertised: 0,            32768
Window scale factor:    0,              0
Window frequency:       0,              0 Windows/s
Time Stamp updates:     0,              0
Total Round Trip Time:  0.000000,        0.000000 ( 0%), ( 0%)
Average Round Trip Time: 0.000000,        0.000000
6 Number of:
Inbound,              Outbound
Packets:               1,              1
(x) Untraced Packets:  0,              0
(.) In-order data:      0,              0 ( 0.00%), ( 0.00%)
(a) Acknowledgments:    1,              0 (100.00%), ( 0.00%)
(+) Data and ACK:       0,              0 ( 0.00%), ( 0.00%)
(u) Duplicate ACKs:     0,              0 ( 0.00%), ( 0.00%)
(w) Window size updates: 0,              0 ( 0.00%), ( 0.00%)
(z) Zero window sizes:  0,              0 ( 0.00%), ( 0.00%)
(p) Window probes:      0,              0 ( 0.00%), ( 0.00%)
(k) Keepalive segments: 0,              0 ( 0.00%), ( 0.00%)
(r) Retransmissions:    0,              0 ( 0.00%), ( 0.00%)
(o) Out-of-order:       0,              0 ( 0.00%), ( 0.00%)
(d) Delayed ACKs:       0,              0 ( 0.00%), ( 0.00%)
(f) Fragments:          0,              0 ( 0.00%), ( 0.00%)
7 Time Spent on:
Inbound,              Outbound
(.) In-order data:      00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
(a) Acknowledgments:    00:00:00.000957, 00:00:00.000000 (106.33%), ( 0.00%)
(+) Data and ACK:       00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
(u) Duplicate ACKs:     00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
(w) Window size updates: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
(z) Zero window sizes:  00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
```

```

(p) Window probes: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
(k) Keepalive segments: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
(r) Retransmissions: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
(o) Out-of-order: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
(d) Delayed ACKs: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
(f) Fragments: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
8 Number of: Inbound, Outbound
  ( S ) SYN: 0, 1 ( 0.00%), (100.00%)
  ( A S ) ACK SYN: 0, 0 ( 0.00%), ( 0.00%)
  ( F ) FIN: 0, 0 ( 0.00%), ( 0.00%)
  ( A F ) ACK FIN: 0, 0 ( 0.00%), ( 0.00%)
  ( R ) RST: 1, 0 (100.00%), ( 0.00%)
  ( U ) URG: 0, 0 ( 0.00%), ( 0.00%)
9 Time Spent on: Inbound, Outbound
  ( S ) SYN: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  ( A S ) ACK SYN: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  ( F ) FIN: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  ( A F ) ACK FIN: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  ( R ) RST: 00:00:00.000957, 00:00:00.000000 (106.33%), ( 0.00%)
  ( U ) URG: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)

```

Messages:

- 2) The largest inbound window is less than twice the inbound MSS.
- 2) This may reduce inbound throughput for bulk data transfers.
- 2) It is usually desirable for the window size to be at twice the MSS.
- 3) The outbound side of the connection appears to be a bulk data transfer.

I - Inbound packet
O - Outbound packet

10	TcpHdr	IO	F	Seq	Ack	RcvWnd	Data	Delta	Time	TimeStamp	RcdNr	State	Inf	Ip_id	Rtt	TimeStmpV	Time
	S	O		29260429	0	32768	0	0.000000	19:55:46.934032		101	SYN_SENT	4	0028	0.00	0.000	316250
	A	R	I	a	0	29260430	0	0.000957	19:55:46.934989		102	CLOSED	4	0014	0.00	0.000	316250

1

Host

The number of packets records for this session; the IP addresses and port of the session.

2

Connection

The first and last time of the session, the length of the session, the final value of RRT, and the final state of the session.

3

Data Quality & Throughput

These statistics are about the quantity of data transmitted. The number of bytes received inbound and the number of bytes send outbound.

4

Data Segment Stats

These statistics are about the segments, the number, and the sizes of the segments. The maximum segment size is captured from the SYN packet.

5

Window Stats

These statistics are about the window changes. The Window scale factor is captured from the SYN packet. The Time Stamp updates are captured from the Tcp header options.

6

Number of Packets

These statistics are about the number of data packets that flow for carrying data. The percentages are based on the number of packets.

7

Time Spent on:

These statistics are about the delta times of data packets that flow for carrying data. The percentages are based on the duration of the session.

8

Number of

These statistics are about the number of control packets that flow for starting and ending a session. The percentages are based on the number of packets.

9

Time Spent on

These statistics are about the delta times of control packets that flow for starting and ending a session. The percentages are based on the duration of the session.

10

Details TcpHdr

The flags from the TCP header

- *** This packet has been reassembled.
- A** This packet is an acknowledgment.
- P** This packet has the PUSH flag set.
- U** This packet is urgent.
- S** This packet is a syn.
- F** This packet is a fin.
- R** This packet is a reset.

The type of data packet has one of the following flags:

- .** The packet flowed in order with respect to its sequence number.
- x** There is a gap in the sequence number and there appears to be untraced data.
- a** The packet is a stand-alone acknowledgment of previously received data.
- +** The packet is an acknowledgment of previously received data and also contains data.
- u** The packet is an acknowledgment of data previously acknowledged.
- w** The packet updated the window size.
- z** The packet changed the window size to zero.
- p** The packet was a window probe.
- k** The packet was a keepalive packet.
- r** The packet was retransmission.
- o** The packet arrived out of order.
- d** The packet exceeded the delay time threshold.
- f** The packet was a fragment of a complete IP packet.

SNIFFER:

Purpose: This report shows information written to the SNIFFER data set.

Format:

```
ALLOC F(SNIFFER) DATASET(SNIFFER.TRC) LRECL(1600) RECFM(V B) REUSE TRACK SPACE(15 15)
CTTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((SNIFFER NOREASSEMBLY STATS))
```

Examples:

Interface Table Report

1	Index	Count	Link	Address
	1	5	OSAQDIOLINK	9.67.115.63
	2	42	LOOPBACK	127.0.0.1
	3	18	OSAQDIOLINK	9.67.115.5
	4	31	OSAQDI046	FEC9:C2D4::6:2900:EDC:217C
	5	21	OSAQDI046	FE80::6:2900:EDC:217C
	6	6	LOOPBACK6	::1

=====

Sniffer Report

2	126 records written to USER2.SNIFFER.TRC
3	13,982 bytes written
4	0 packets were abbreviated
5	5 packets were truncated to 200 bytes

1

The list of device names found in the selected records. Each device is assigned an interface index.

2

This record count includes the two header records and one trailer record written to the SNIFFER data set.

3

The number of data bytes written to the SNIFFER data set. This is the amount of data to be downloaded.

4

The number of abbreviated records. This number is included in **5**.

5

The number of truncated records. Records were truncated because the size of the packet exceeded the logical record length of the SNIFFER file. You can avoid this by increasing the logical record length. The maximum logical record length is 32,763 or the size of physical disk blocks, whichever is smaller.

STATISTICS:

Purpose: The records are counted by record type, device type, device name, job name, protocol, IP address, TCP port number, and UDP port number.

Format:

CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((STATISTICS(DETAIL)))

Examples:

1 No packets required reassembly

SYSTCPDA Trace Statistics

```
123 ctrace records processed
0 segmented trace records read
0 segmented trace records were lost
123 trace records read
0 records could not be validated
123 records passed filtering
123 packet trace records processed
0 data trace records processed
```

2 Record Type Report

Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Record Type
65	55	3814	10	644	1	2002/02/12	19:49:42	123	2002/02/12	19:57:45	1(Packet Trace)
58	19	1828	39	2840	40	2002/02/12	19:52:39	117	2002/02/12	19:56:29	4(Packet Trace)
123	74	5642	49	3484	Total						

2 Record Type(s) found

Ip Version Report

Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Ip Version
65	55	3814	10	644	1	2002/02/12	19:49:42	123	2002/02/12	19:57:45	4
58	19	1828	39	2840	40	2002/02/12	19:52:39	117	2002/02/12	19:56:29	6
123	74	5642	49	3484	Total						

2 Ip Version(s) found

3 Device Type Report

Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Device Type
42	42	2667	0	0	4	2002/02/12	19:49:48	123	2002/02/12	19:57:45	34(Loopback)
23	13	1147	10	644	1	2002/02/12	19:49:42	103	2002/02/12	19:55:48	39(QDIO Ethernet)
6	3	324	3	180	49	2002/02/12	19:52:58	54	2002/02/12	19:52:58	51(Loopback6)
52	16	1504	36	2660	40	2002/02/12	19:52:39	117	2002/02/12	19:56:29	53(QDIO Ethernet6)
123	74	5642	49	3484	Total						

4 Device Type(s) found

4 Interface Report

Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Interface
42	42	2667	0	0	4	2002/02/12	19:49:48	123	2002/02/12	19:57:45	LOOPBACK
6	3	324	3	180	49	2002/02/12	19:52:58	54	2002/02/12	19:52:58	LOOPBACK
23	13	1147	10	644	1	2002/02/12	19:49:42	103	2002/02/12	19:55:48	OSAQDIOL
52	16	1504	36	2660	40	2002/02/12	19:52:39	117	2002/02/12	19:56:29	OSAQDI04
123	74	5642	49	3484	Total						

4 Interface(s) found

Interface Address Report

Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Interface
5	5	699	0	0	1	2002/02/12	19:49:42	103	2002/02/12	19:55:48	OSAQDIOLINK
											Addr: 9.67.115.63
42	42	2667	0	0	4	2002/02/12	19:49:48	123	2002/02/12	19:57:45	LOOPBACK
											Addr: 9.67.115.5
18	8	448	10	644	16	2002/02/12	19:51:17	33	2002/02/12	19:52:21	OSAQDIOLINK
											Addr: 9.67.115.5
31	14	1360	17	1340	40	2002/02/12	19:52:39	116	2002/02/12	19:56:29	OSAQDI046
											Addr: FEC9:C2D4::6:2900:EDC:217C
21	2	144	19	1320	46	2002/02/12	19:52:44	117	2002/02/12	19:56:29	OSAQDI046
											Addr: FE80::6:2900:EDC:217C
6	3	324	3	180	49	2002/02/12	19:52:58	54	2002/02/12	19:52:58	LOOPBACK6
											Addr: FEC9:C2D4::9:67:115:5
123	74	5642	49	3484	Total						

6 Interface Address(s) found

Asid Report

Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Asid
34	0	0	34	2440	16	2002/02/12	19:51:17	111	2002/02/12	19:56:24	002A
89	74	5642	15	1044	1	2002/02/12	19:49:42	123	2002/02/12	19:57:45	002B
123	74	5642	49	3484	Total						

2 Asid(s) found

5 Protocol Report

Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Protocol
30	29	1624	1	284	5	2002/02/12	19:49:48	123	2002/02/12	19:57:45	1(ICMP)
4	2	120	2	160	99	2002/02/12	19:55:20	102	2002/02/12	19:55:46	6(TCP)
53	26	2190	27	1440	1	2002/02/12	19:49:42	122	2002/02/12	19:57:45	17(UDP)
36	17	1708	19	1600	41	2002/02/12	19:52:39	117	2002/02/12	19:56:29	58(ICMPv6)

123	74	5642	49	3484	Total										
4 Protocol(s) found															
6	IP Address Report														
Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss					
9	5	280	4	404	16	2002/02/12	19:51:17	27	2002/02/12	19:52:21					
						Addr: 9.67.115.1									
60	50	3115	10	644	4	2002/02/12	19:49:48	123	2002/02/12	19:57:45					
						Addr: 9.67.115.5									
5	5	699	0	0	1	2002/02/12	19:49:42	103	2002/02/12	19:55:48					
						Addr: 9.67.115.17									
5	5	699	0	0	1	2002/02/12	19:49:42	103	2002/02/12	19:55:48					
						Addr: 9.67.115.63									
9	3	168	6	240	22	2002/02/12	19:52:21	33	2002/02/12	19:52:21					
						Addr: 9.67.115.69									
4	0	0	4	240	55	2002/02/12	19:53:17	67	2002/02/12	19:53:32					
						Addr: FE80::6:2900:ADC:217C									
21	2	144	19	1320	46	2002/02/12	19:52:44	117	2002/02/12	19:56:29					
						Addr: FE80::6:2900:EDC:217C									
4	2	144	2	144	45	2002/02/12	19:52:44	105	2002/02/12	19:55:51					
						Addr: FE80::202:55FF:FE64:2DE7									
2	1	72	1	72	116	2002/02/12	19:56:29	117	2002/02/12	19:56:29					
						Addr: FE80::206:2AFF:FE66:C800									
6	3	216	3	216	76	2002/02/12	19:54:02	94	2002/02/12	19:55:09					
						Addr: FE80::206:2AFF:FE71:4400									
31	14	1360	17	1340	40	2002/02/12	19:52:39	116	2002/02/12	19:56:29					
						Addr: FEC9:C2D4::6:2900:EDC:217C									
6	3	324	3	180	49	2002/02/12	19:52:58	54	2002/02/12	19:52:58					
						Addr: FEC9:C2D4::9:67:115:5									
8	4	348	4	260	40	2002/02/12	19:52:39	102	2002/02/12	19:55:46					
						Addr: FEC9:C2D4::9:67:115:17									
3	2	376	1	304	110	2002/02/12	19:56:24	113	2002/02/12	19:56:24					
						Addr: FEC9:C2D4::206:2AFF:FE66:C800									
8	4	348	4	260	88	2002/02/12	19:55:04	100	2002/02/12	19:55:20					
						Addr: FEC9:C2D4::206:2AFF:FE71:4400									
5	0	0	5	300	74	2002/02/12	19:53:57	82	2002/02/12	19:54:17					
						Addr: FEC9:C2D4:1::9:67:114:44									
1	0	0	1	72	41	2002/02/12	19:52:39	41	2002/02/12	19:52:39					
						Addr: FF02::1:FF15:17									
1	0	0	1	72	111	2002/02/12	19:56:24	111	2002/02/12	19:56:24					
						Addr: FF02::1:FF66:C800									
1	0	0	1	72	89	2002/02/12	19:55:04	89	2002/02/12	19:55:04					
						Addr: FF02::1:FF71:4400									
9	0	0	9	648	56	2002/02/12	19:53:17	66	2002/02/12	19:53:31					
						Addr: FF02::1:FFDC:217C									
198	103	8293	95	6788	Total										
20 IP Address(s) found															
Qos Report															
Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Qos				
5	5	280	0	0	18	2002/02/12	19:51:54	27	2002/02/12	19:52:21	6	Internetwork)			
1	1	60	0	0	100	2002/02/12	19:55:20	100	2002/02/12	19:55:20	96	(Unknown)			
6	6	432	0	0	77	2002/02/12	19:54:02	116	2002/02/12	19:56:29	112	(Unknown)			
12	12	772	0	0	Total										
3 Qos(s) found															
7	Tcp Port Report														
Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Tcp Port				
4	2	120	2	160	99	2002/02/12	19:55:20	102	2002/02/12	19:55:46	21	(ftp)			
2	1	60	1	80	99	2002/02/12	19:55:20	100	2002/02/12	19:55:20	1026	()			
2	1	60	1	80	101	2002/02/12	19:55:46	102	2002/02/12	19:55:46	1027	()			
8	4	240	4	320	Total										
3 Tcp Port(s) found															
8	Udp Port Report														
Total	Input	Data	Output	Data	First	yyyy/mm/dd	hh.mm.ss	Last	yyyy/mm/dd	hh.mm.ss	Udp Port				
3	3	234	0	0	1	2002/02/12	19:49:42	3	2002/02/12	19:49:44	137	(netbios-ns)			
2	2	465	0	0	83	2002/02/12	19:54:38	103	2002/02/12	19:55:48	138	(netbios-dgm)			
21	21	1491	0	0	4	2002/02/12	19:49:48	122	2002/02/12	19:57:45	161	(snmp)			
3	3	213	0	0	4	2002/02/12	19:49:48	8	2002/02/12	19:49:57	1035	()			
3	3	213	0	0	10	2002/02/12	19:51:06	14	2002/02/12	19:51:15	1036	()			
3	3	213	0	0	34	2002/02/12	19:52:24	38	2002/02/12	19:52:33	1037	()			
3	3	213	0	0	68	2002/02/12	19:53:42	72	2002/02/12	19:53:51	1038	()			
3	3	213	0	0	84	2002/02/12	19:55:00	97	2002/02/12	19:55:09	1039	()			
3	3	213	0	0	106	2002/02/12	19:56:18	114	2002/02/12	19:56:27	1040	()			
3	3	213	0	0	118	2002/02/12	19:57:36	122	2002/02/12	19:57:45	1041	()			
27	0	0	27	1440	17	2002/02/12	19:51:54	95	2002/02/12	19:55:09	32810	()			
7	0	0	7	380	17	2002/02/12	19:51:54	88	2002/02/12	19:55:04	33435	()			
7	0	0	7	380	19	2002/02/12	19:51:54	92	2002/02/12	19:55:04	33436	()			
7	0	0	7	380	20	2002/02/12	19:51:59	95	2002/02/12	19:55:09	33437	()			
3	0	0	3	160	28	2002/02/12	19:52:21	81	2002/02/12	19:54:12	33438	()			
2	0	0	2	100	30	2002/02/12	19:52:21	82	2002/02/12	19:54:17	33439	()			
1	0	0	1	40	32	2002/02/12	19:52:21	32	2002/02/12	19:52:21	33440	()			
101	47	3681	54	2880	Total										
17 Udp Port(s) found															
Protocol Summary Report															

Protocol	Input		Output		Total			
	Packets	Bytes	Packets	Bytes	Packets	Bytes		
Tcp	2	120	2	160	4	280		
Udp	26	2190	27	1440	53	3630		
Icmp	46	3332	20	1884	66	5216		
Other	0	0	0	0	0	0		
Session Summary Report								
Input	Output	First	yyyy/mm/dd	hh:mm:ss	Last	yyyy/mm/dd	hh:mm:ss	Protocol
5	1	16	2002/02/12	19:51:17	27	2002/02/12	19:52:21	ICMP
0	1	17	2002/02/12	19:51:54	17	2002/02/12	19:51:54	UDP
								Lc1: 9.67.115.5-32810
0	1	19	2002/02/12	19:51:54	19	2002/02/12	19:51:54	UDP
								Rmt: 9.67.115.1-33435
0	1	20	2002/02/12	19:51:59	20	2002/02/12	19:51:59	UDP
								Lc1: 9.67.115.5-32810
21	0	5	2002/02/12	19:49:48	123	2002/02/12	19:57:45	ICMP
								Rmt: 9.67.115.1-33437
3	0	4	2002/02/12	19:49:48	8	2002/02/12	19:49:57	UDP
								Lc1: 9.67.115.5-0
3	0	10	2002/02/12	19:51:06	14	2002/02/12	19:51:15	UDP
								Rmt: 9.67.115.5-1035
3	0	34	2002/02/12	19:52:24	38	2002/02/12	19:52:33	UDP
								Lc1: 9.67.115.5-161
3	0	68	2002/02/12	19:53:42	72	2002/02/12	19:53:51	UDP
								Rmt: 9.67.115.5-1036
3	0	84	2002/02/12	19:55:00	97	2002/02/12	19:55:09	UDP
								Lc1: 9.67.115.5-161
3	0	106	2002/02/12	19:56:18	114	2002/02/12	19:56:27	UDP
								Rmt: 9.67.115.5-1039
3	0	118	2002/02/12	19:57:36	122	2002/02/12	19:57:45	UDP
								Lc1: 9.67.115.5-161
3	0	29	2002/02/12	19:52:21	33	2002/02/12	19:52:21	ICMP
								Rmt: 9.67.115.5-1041
0	1	22	2002/02/12	19:52:21	22	2002/02/12	19:52:21	UDP
								Lc1: 9.67.115.5-0
0	1	24	2002/02/12	19:52:21	24	2002/02/12	19:52:21	UDP
								Rmt: 9.67.115.69-0
0	1	26	2002/02/12	19:52:21	26	2002/02/12	19:52:21	UDP
								Lc1: 9.67.115.5-32810
0	1	28	2002/02/12	19:52:21	28	2002/02/12	19:52:21	UDP
								Rmt: 9.67.115.69-33436
0	1	30	2002/02/12	19:52:21	30	2002/02/12	19:52:21	UDP
								Lc1: 9.67.115.5-32810
0	1	32	2002/02/12	19:52:21	32	2002/02/12	19:52:21	UDP
								Rmt: 9.67.115.69-33437
3	0	1	2002/02/12	19:49:42	3	2002/02/12	19:49:44	UDP
								Lc1: 9.67.115.5-32810
2	0	83	2002/02/12	19:54:38	103	2002/02/12	19:55:48	UDP
								Rmt: 9.67.115.69-33440
0	1	55	2002/02/12	19:53:17	55	2002/02/12	19:53:17	UDP
								Lc1: 9.67.115.63-137
0	1	59	2002/02/12	19:53:22	59	2002/02/12	19:53:22	UDP
								Rmt: 9.67.115.17-137
0	1	63	2002/02/12	19:53:27	63	2002/02/12	19:53:27	UDP
								Lc1: 9.67.115.63-138
0	1	67	2002/02/12	19:53:32	67	2002/02/12	19:53:32	UDP
								Rmt: 9.67.115.17-138
0	2	46	2002/02/12	19:52:44	105	2002/02/12	19:55:51	ICMPv6
								Lc1: FE80::6:2900:EDC:217C-32810
0	1	117	2002/02/12	19:56:29	117	2002/02/12	19:56:29	ICMPv6
								Rmt: FE80::6:2900:ADC:217C-33435
2	3	76	2002/02/12	19:54:02	94	2002/02/12	19:55:09	ICMPv6
								Lc1: FE80::6:2900:EDC:217C-32810
0	9	56	2002/02/12	19:53:17	66	2002/02/12	19:53:31	ICMPv6
								Rmt: FE80::6:2900:ADC:217C-33437
2	0	45	2002/02/12	19:52:44	104	2002/02/12	19:55:51	ICMPv6
								Lc1: FE80::202:55FF:FE64:2DE7-0
1	0	116	2002/02/12	19:56:29	116	2002/02/12	19:56:29	ICMPv6
								Rmt: FE80::206:2AFF:FE66:C800-0
1	0	93	2002/02/12	19:55:09	93	2002/02/12	19:55:09	ICMPv6
								Lc1: FE80::6:2900:EDC:217C-0
1	1	101	2002/02/12	19:55:46	102	2002/02/12	19:55:46	TCP
								Rmt: FE80::206:2AFF:FE71:4400-0
0	1	40	2002/02/12	19:52:39	40	2002/02/12	19:52:39	UDP
								Lc1: FEC9:C2D4::6:2900:EDC:217C-1027
0	1	44	2002/02/12	19:52:39	44	2002/02/12	19:52:39	UDP
								Rmt: FEC9:C2D4::9:67:115:17-21
0	1	47	2002/02/12	19:52:44	47	2002/02/12	19:52:44	UDP
								Lc1: FEC9:C2D4::9:67:115:17-33435
3	0	42	2002/02/12	19:52:39	48	2002/02/12	19:52:44	ICMPv6
								Rmt: FEC9:C2D4::6:2900:EDC:217C-32810

2	1	110	2002/02/12 19:56:24	113	2002/02/12 19:56:24	ICMPv6	Lc1: FEC9:C2D4::6:2900:EDC:217C-0 Rmt: FEC9:C2D4::206:2AFF:FE66:C800-0
1	1	99	2002/02/12 19:55:20	100	2002/02/12 19:55:20	TCP	Lc1: FEC9:C2D4::6:2900:EDC:217C-1026 Rmt: FEC9:C2D4::206:2AFF:FE71:4400-21
0	1	88	2002/02/12 19:55:04	88	2002/02/12 19:55:04	UDP	Lc1: FEC9:C2D4::6:2900:EDC:217C-32810 Rmt: FEC9:C2D4::206:2AFF:FE71:4400-33435
0	1	92	2002/02/12 19:55:04	92	2002/02/12 19:55:04	UDP	Lc1: FEC9:C2D4::6:2900:EDC:217C-32810 Rmt: FEC9:C2D4::206:2AFF:FE71:4400-33436
0	1	95	2002/02/12 19:55:09	95	2002/02/12 19:55:09	UDP	Lc1: FEC9:C2D4::6:2900:EDC:217C-32810 Rmt: FEC9:C2D4::206:2AFF:FE71:4400-33437
3	0	90	2002/02/12 19:55:04	96	2002/02/12 19:55:09	ICMPv6	Lc1: FEC9:C2D4::6:2900:EDC:217C-0 Rmt: FEC9:C2D4::206:2AFF:FE71:4400-0
0	1	74	2002/02/12 19:53:57	74	2002/02/12 19:53:57	UDP	Lc1: FEC9:C2D4::6:2900:EDC:217C-32810 Rmt: FEC9:C2D4:1::9:67:114:44-33435
0	1	75	2002/02/12 19:54:02	75	2002/02/12 19:54:02	UDP	Lc1: FEC9:C2D4::6:2900:EDC:217C-32810 Rmt: FEC9:C2D4:1::9:67:114:44-33436
0	1	78	2002/02/12 19:54:07	78	2002/02/12 19:54:07	UDP	Lc1: FEC9:C2D4::6:2900:EDC:217C-32810 Rmt: FEC9:C2D4:1::9:67:114:44-33437
0	1	81	2002/02/12 19:54:12	81	2002/02/12 19:54:12	UDP	Lc1: FEC9:C2D4::6:2900:EDC:217C-32810 Rmt: FEC9:C2D4:1::9:67:114:44-33438
0	1	82	2002/02/12 19:54:17	82	2002/02/12 19:54:17	UDP	Lc1: FEC9:C2D4::6:2900:EDC:217C-32810 Rmt: FEC9:C2D4:1::9:67:114:44-33439
0	1	41	2002/02/12 19:52:39	41	2002/02/12 19:52:39	ICMPv6	Lc1: FEC9:C2D4::6:2900:EDC:217C-0 Rmt: FF02::1:FF15:17-0
0	1	111	2002/02/12 19:56:24	111	2002/02/12 19:56:24	ICMPv6	Lc1: FEC9:C2D4::6:2900:EDC:217C-0 Rmt: FF02::1:FF66:C800-0
0	1	89	2002/02/12 19:55:04	89	2002/02/12 19:55:04	ICMPv6	Lc1: FEC9:C2D4::6:2900:EDC:217C-0 Rmt: FF02::1:FF71:4400-0
0	1	49	2002/02/12 19:52:58	49	2002/02/12 19:52:58	UDP	Lc1: FEC9:C2D4::9:67:115:5-32810 Rmt: FEC9:C2D4::9:67:115:5-33435
0	1	51	2002/02/12 19:52:58	51	2002/02/12 19:52:58	UDP	Lc1: FEC9:C2D4::9:67:115:5-32810 Rmt: FEC9:C2D4::9:67:115:5-33436
0	1	53	2002/02/12 19:52:58	53	2002/02/12 19:52:58	UDP	Lc1: FEC9:C2D4::9:67:115:5-32810 Rmt: FEC9:C2D4::9:67:115:5-33437
3	0	50	2002/02/12 19:52:58	54	2002/02/12 19:52:58	ICMPv6	Lc1: FEC9:C2D4::9:67:115:5-0 Rmt: FEC9:C2D4::9:67:115:5-0

9 55 session(s) found
 10 123 records processed for this report
 11 Recording ended at 2002/02/12 19:57:45.836155
 Recording started at 2002/02/12 19:49:42.788207
 The duration was 00:08:03.047947
 12 304 is the maximum packet data length
 13 16384 bytes of storage used to create this report
 123 requests for 14992 bytes of storage were made

1

The standard statistics shown with all executions of the SYSTCPDA packet trace formatter.

Ctrace records processed

The total number of Ctrace records given to the SYSTCPDA packet trace formatter by IPCS.

segmented trace records read

The total number of packets that spanned multiple Ctrace records.

segmented trace records were lost

The total number of segmented packets records that could not be put back together.

trace records read

The total number of complete trace records.

records could not be validated

The number of incomplete Ctrace records that could not be used.

records passed filtering

The number of records that were successfully formatted.

packet trace records processed

The number of records that were packet trace records.

data trace records processed

The number of records that were data trace records.

- 2** The totals by record type (Packet trace, X25, and data trace).
- 3** The totals by device type for packet trace records.
- 4** The totals by Interface or Link Name for packet trace records.
- 5** The totals by Protocol number for packet trace records.
- 6** The totals by IP Address. Both the destination and source IP addresses are counted except when they are the same within a record.
- 7** The totals by TCP Port number. Both the destination and source port numbers are counted except when they are the same within a record.
- 8** The totals by UDP port number. Both the destination and source port numbers are counted except when they are the same within a record.
- Note:** Reports **2** through **8** are shown only when STATISTICS(DETAIL) is specified in the OPTIONS string.
- 9** The totals by session partner pairs (IP addresses, protocol number, and port numbers).
- 10** The number of records processed for the statistics report.
- 11** The time stamp of the first record in the input file, the time stamp of the last record in the input, and the duration from the first to last record.
- Note:** Not shown in this example are records that have been abbreviated. The number of records that were abbreviated and the maximum abbreviated size are shown. Also the number and maximum size of the records that were not abbreviated are shown.
- 12** The size of the largest packet found in the input file.
- 13** The number of records processed for the statistics report, the number of 1KB blocks of storage required for this report, the number of storage requests, and the total amount of storage required for the requests.
- The report by Jobname for data trace records is not shown. Each category of totals is broken down by:
- The total number of records
 - The total number of inbound records
 - The total amount of inbound data
 - The total number of outbound records
 - The total amount of outbound data
 - The record number of the first record
 - The time stamp of the first record

- The record number of the last record
- The time stamp of the last record

STREAM:

Purpose: Sometimes messages span multiple packets. TELNET and DNS are examples. The STREAM report (with the DUMP or FORMAT keywords) capture the entire stream of data.

Format:

```
CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((STREAM DUMP ASCII))
```

Examples:

```
=====
1 Streams Report
  60 Streams found
  23600 bytes of storage for the session report was allocated
  1146880 bytes of storage for buffers was allocated

-----
2 Session: FEC9:C2D4::206:2AFF:FE71:4400-0 FEC9:C2D4::6:2900:EDC:217C-0 ICMPv6
  From: 2002/02/12 19:55:04.615118 to: 2002/02/12 19:55:09.636234
  3 packets found
  Stream buffer at 0A818000 for 20480 bytes. 144 bytes were used
  3 packets moved for 144 bytes
  I - Inbound packet
  O - Outbound packet

3 D Rcd #           Time           Delta           Seq #    Position Length    End_Pos
I   90 19:55:04.615118 00:00:00.000000          0         0      24       24
000000 FEC9C2D4 00000000 02062AFF FE714400 02010006 2A714400 |.....*..qD.....*qD. |
I   91 19:55:04.631206 00:00:00.016087          24        24      60      84
000000                                60000000 00141101 |.....~..... |
000020 FEC9C2D4 00000000 00062900 0EDC217C FEC9C2D4 00000000 02062AFF FE714400 |.....)....!|.....*..qD. |
000040 802A829B 0014A3B6 01010000 3C697318 00095CAB |.....<is..... |
```

1

After all the records have processed, the number of streams and the amount of storage required for the report and buffers are shown.

2

Each session is identified by the IP addresses, port number, and protocol. The time stamps of the first and last packet are shown along with the number of packets, the address, and size of the stream buffer.

3

When a stream is dumped, each packet and the data from the packet is shown. If there are gaps in the stream, the number of bytes skipped will be displayed. The data about each packet formatted are:

D The direction of the packet: I for inbound and O for outbound.

Rcd

The record number.

Time

The time stamp of the record.

Delta

The time from the first record of the stream.

Seq

The sequence number of the TCP packet. For other packets it is the relative offset of the packet from the first packet.

Position

The relative offset of the packet.

Length

The number of bytes in the packet.

End_Pos

The ending sequence number.

Data trace (SYSTCPDA) for TCP/IP stacks

You can use the DATTRACE command to trace socket data (transforms) into and out of the physical file structure (PFS). DATTRACE operates with the following APIs:

- REXX
- C-sockets
- IMS
- CICS
- Native z/OS UNIX
- Macro
- CALL Instruction

For information about the format of the data trace command (VARY DATTRACE) see the *z/OS Communications Server: IP System Administrator's Commands*.

Starting data trace

You can start data trace for all job names using the VARY command:

```
V TCPIP,tcpprocname,DAT
```

Security note:

1. To use any VARY command, the user must be authorized in RACF. This replaces the old OBEY list authorization.
2. Each user's RACF profile must have access for a resource of the form MVS.VARY.TCPIP.xxx, where xxx is the first eight characters of the command name. For data trace, this would be MVS.VARY.TCPIP.DATTRACE.
3. Traces are placed in an internal buffer, which can then be written out using an external writer. The MVS TRACE command must also be issued for component SYSTCPDA to activate the data trace. For details, see Chapter 5, "TCP/IP services traces and IPCS support" on page 53.

Displaying data traces

You can use the NETSTAT or onetstat command to display data traces. Figure 12 shows a data trace for a single entry.

```
onetstat -p TCPCS -f
...
Data Trace Setting:
JobName: *          TrRecCnt: 00000006  Length: FULL
IpAddr:  *          SubNet:  *
```

Figure 12. Data trace: Single entry

Figure 13 on page 136 shows a data trace for multiple entries.

```

onetstat -p TCPCS -f
...
Data Trace Setting:
JobName: MEGA4      TrRecCnt: 00000000 Length: FULL
IpAddr: 127.0.0.3   SubNet: *

JobName: *          TrRecCnt: 00000000 Length: FULL
IpAddr: 127.0.0.9   SubNet: *

```

Figure 13. Data trace: Multiple entries

Formatting data traces using IPCS

Data trace records are written to the same CTRACE component as packet trace records (SYSTCPDA). Thus, all the IPCS formatting features for packet trace are also available for data trace. You may use the ENTIDLIST parameter to isolate data trace records and packet trace records from each other. See the “Formatting packet traces using IPCS” on page 95 section for more information.

Intrusion Detection Services trace (SYSTCPIS)

When starting the TCP stack, the stack reads the CTIIDS00 parmlib member to determine the size to reserve for the SYSTCPIS Ctrace. You can override this default by starting TCP/IP with the PARM option and the keyword IDS=xx, where xx is the suffix of the CTIIDSxx PARMLIB member. In the following example, the trace searches for PARMLIB member CTIIDSA3.

```
S tcpiproc,PARM='IDS=A3'
```

If the parmlib member is not found or the member contains data that is not valid, the following message is displayed.

```
EZZ4210I CTRACE DEFINE FAILED FOR CTIIDS00
```

If the EZZ4210I message indicates the parmlib member name CTIIDS00, the IDS CTrace space is set up using the default BUFSIZE of 32M.

The CTIIDS00 member is used to specify the IDS CTrace parameters. To eliminate this message, ensure that a CTIIDS00 member exists within Parmlib and that the options are correctly specified. A sample CTIIDS00 member is shipped with z/OS Communications Server.

Packets are traced based on IDS policy defined in LDAP. Refer to Intrusion Detection Services in the *z/OS Communications Server: IP Configuration Guide* for information on defining policy.

See Chapter 23, “Diagnosing intrusion detection problems” on page 575 for additional information about diagnosing policy problems..

```

/*****
/*
/* Communications Server for OS/390
/* SMP/E Distribution Name: CTIIDS00
/*
/* MEMBER: CTIIDS00
/*
/*
/* Copyright:
/* Licensed Materials - Property of IBM
/* 5694-A01
/* (C) Copyright IBM Corp. 2001, 2002
/*
/*
/* Status: CSV1R4
/*
/*
/* DESCRIPTION = This parmlib member causes IDS component trace
/* for the TCP/IP product to be initialized with a
/* trace buffer size of 128 megabytes.
/*
/* This parmlib members only lists those TRACEOPTS
/* value specific to TCP/IP. For a complete list
/* of TRACEOPTS keywords and their values see
/* OS/390 MVS INITIALIZATION AND TUNING REFERENCE.
/*
/* $MAC(CTIIDS00) PROD(TCPIP): Component Trace SYS1.PARMLIB member
/*
/*
/*
*****/
TRACEOPTS
/* ----- */
/* ON OR OFF: PICK 1 */
/* ----- */
/* ON
/* OFF */
/* ----- */
/* BUFSIZE: A VALUE IN RANGE 16M TO 256M */
/* ----- */
/* BUFSIZE(32M)
/* WTR(wtr_procedure) WRAP|NOWRAP */

```

Figure 14. SYS1.PARMLIB member CTIIDS00

Restrictions

For IDS trace records the COMP keyword must be SYSTCPIS. Because there are no EXCEPTION records for IDS trace, the EXCEPTION keyword must not be specified.

CTRACE keywords on SYSTCPIS

The following describes those CTRACE keywords that affect SYSTCPIS processing.

ENTIDLIST

The ENTIDLIST keyword can be used to select trace records with a specific Probeld.

JOBLIST, JOBNAME

The JOBLIST and JOBNAME keywords can be used to select trace records with a matching job name. The JOBNAME keyword in the OPTIONS list can also be used to select records.

ASIDLIST

The ASIDLIST can be used to select trace records with a matching Asid.

GMT

The time stamps are converted to GMT time.

LOCAL

The time stamps are converted to LOCAL time.

SHORT

If the OPTIONS keyword does not specify any reports, format the trace records. Equivalent to the FORMAT option.

FULL

If the OPTIONS keyword does not specify any reports, format and dump the trace records. Equivalent to the FORMAT and DUMP options.

SUMMARY

If the OPTIONS keyword does not specify any reports, create a one line summary for each trace record. Equivalent to the SUMMARY option.

TALLY

If the OPTIONS keyword does not specify any reports, then count the trace records. Equivalent to the STATISTICS option.

START and STOP

These keywords limit the trace records seen by the packet trace formatter. The START keyword determines the time when records are seen by the packet trace report formatter. The STOP keyword determines the time when records are no longer seen by the packet trace report formatter.

Note: CTRACE always uses the time the trace record was moved to the buffer for START and STOP times.

LIMIT

Determines the number of records that the packet trace formatter is allowed to process. See the RECORDS keyword value in OPTIONS.

USEREXIT

The CTRACE USEREXIT is not called because the packet trace formatter tells CTRACE to skip all the records. Therefore, the packet trace formatter calls the CTRACE USEREXIT before testing the records with the filtering criteria. If it returns a nonzero return code, the record is skipped. The USEREXIT can also be used in the OPTIONS keyword. It is called after the record has met all the filtering criteria in the OPTIONS keyword. The OPTIONS keyword provides a means of entering additional keywords for record selection and formatting.

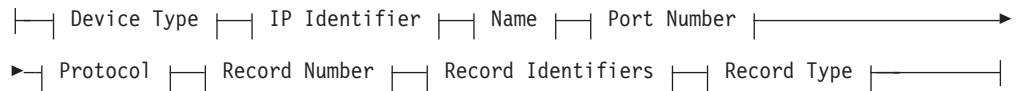
SYSTCPIS OPTIONS

The syntax for the OPTIONS component routine parameters is:

OPTIONS component

►►—OPTIONS—((—| Data Selection |—| Report Generation |—))—►►

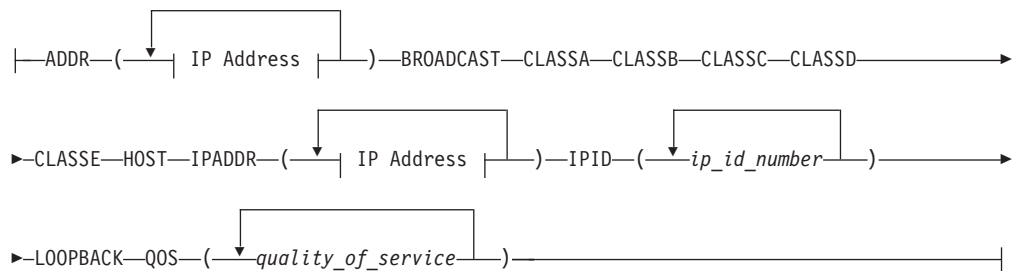
Data Selection:



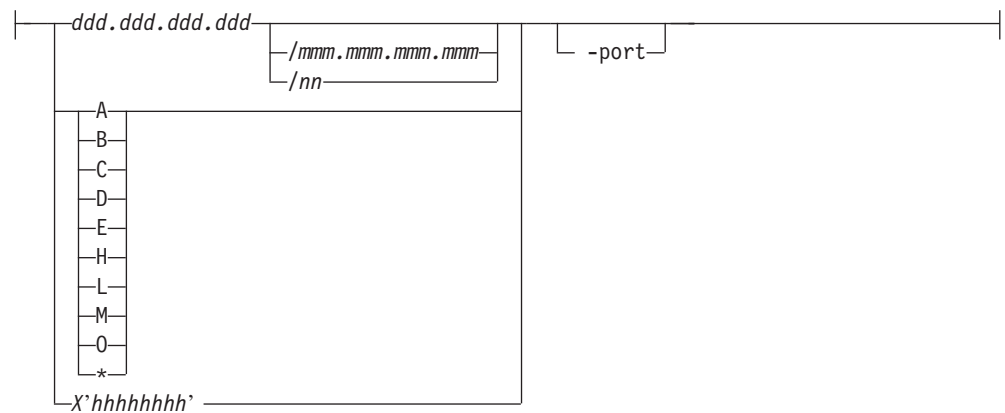
Device Type:



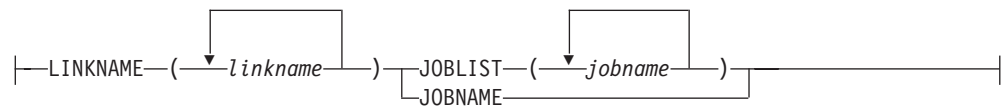
IP Identifier:



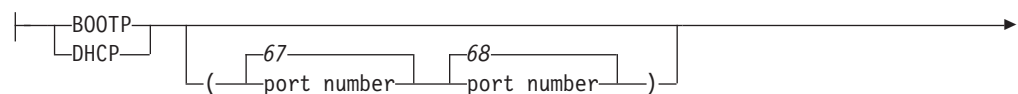
IP Address:

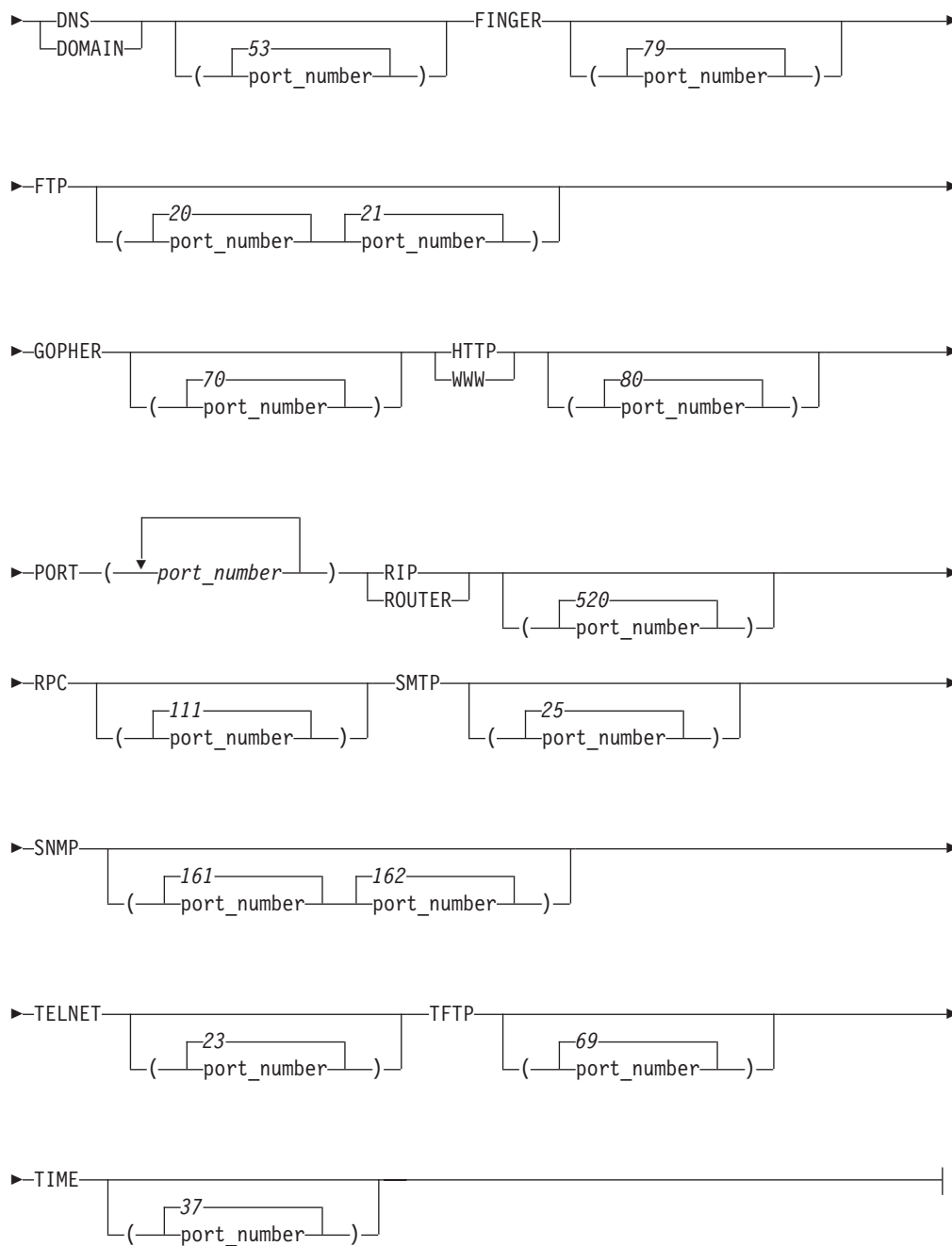


Name:

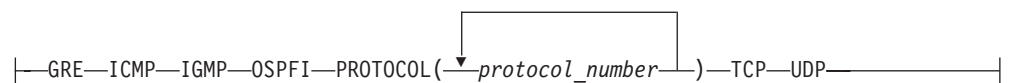


Port Number:





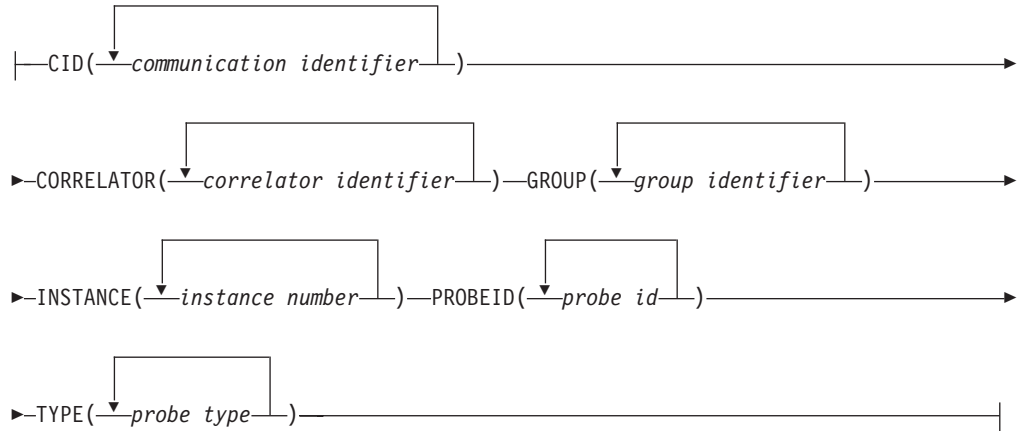
Protocol:



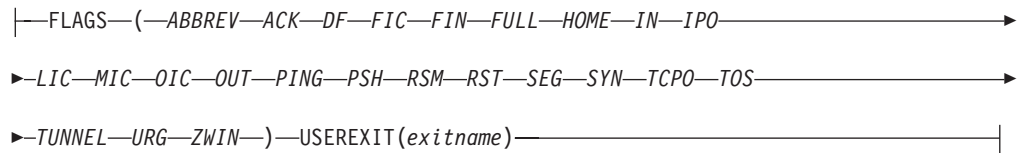
Record Number:



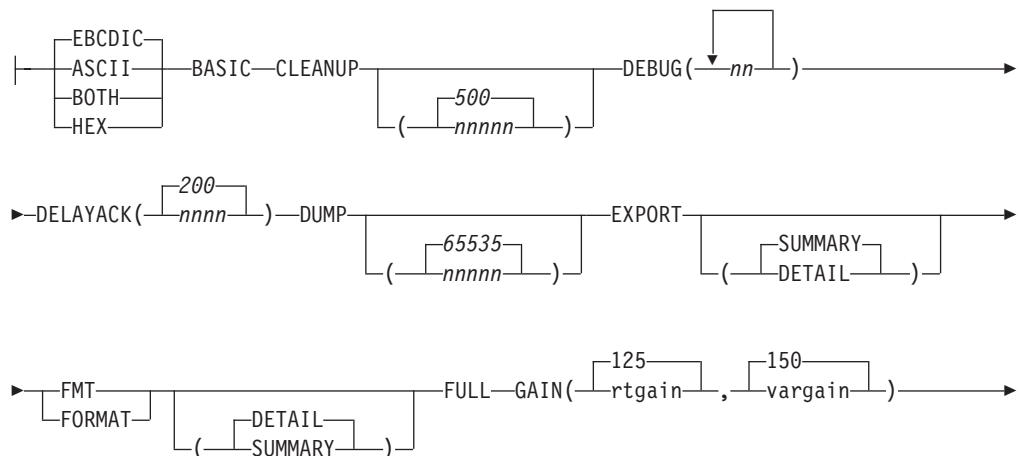
Record Identifiers:

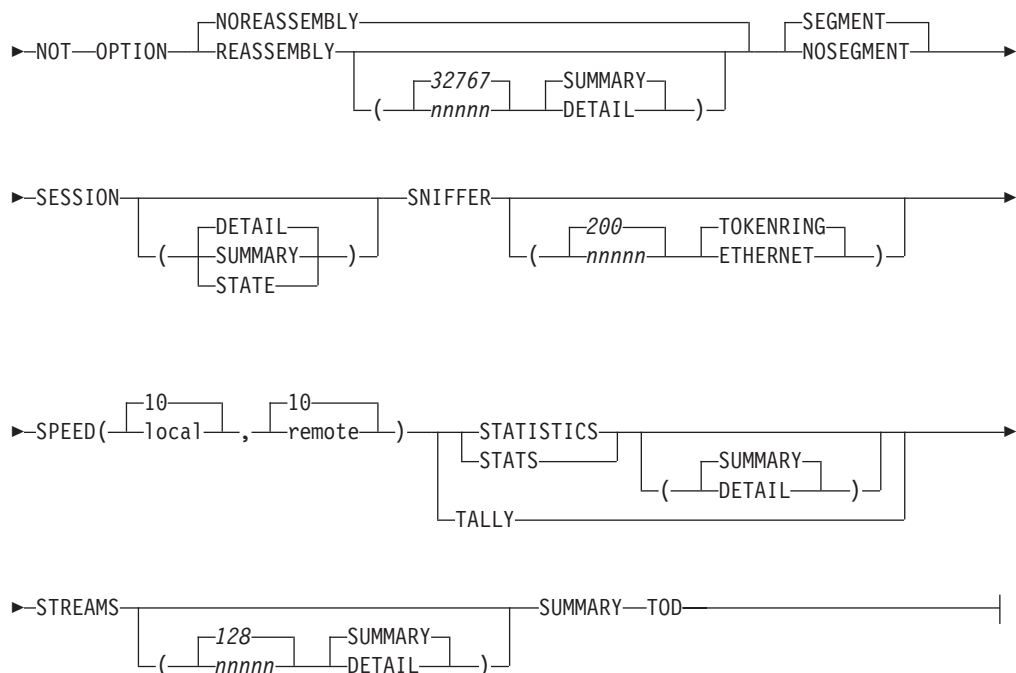


Record Type:



Report Generation:





OPTIONS Keywords

The following are keywords used for the OPTIONS component routine parameters.

ASCII

Dump trace record data with ASCII translation only. The default is BOTH.

BASIC

For specific packet types dump each element of the packet. Applies to DNS, RIP, and SNMP packet data.

BOOTP[(port_number|67; port_number|68)]

Select BOOTP protocol packets. The port_number defines the BOOTP port numbers to select trace records for formatting. Equivalent to PORT(67 68).

BOTH

Dump trace record data with both ASCII and EBCDIC translations. This is the default.

BROADCAST

Select trace records with a broadcast IP address. Equivalent to IPADDR(255.255.255.255/255.255.255.255).

CID

A connection identifier. Up to 16 identifiers can be specified. The CID values can be entered in either decimal (such as CID(182)) or hexadecimal (such as CID(X'0006CE7E')), but are displayed in hexadecimal. This is the same value that appears in the NETSTAT connections display.

CLASSA

Select trace records with a class A IP address. Equivalent to IPADDR(0.0.0.0/128.0.0.0).

CLASSB

Select trace records with a class B IP address. Equivalent to IPADDR(128.0.0.0/192.0.0.0).

CLASSC

Select trace records with a class C IP address. Equivalent to IPADDR(192.0.0.0/224.0.0.0).

CLASSD

Select trace records with a class D IP address. Equivalent to IPADDR(224.0.0.0/240.0.0.0).

CLASSE

Select trace records with a class E IP address. Equivalent to IPADDR(240.0.0.0/248.0.0.0).

CLEANUP(nnnnnl500)

Defines a record interval where saved packet information in storage is released. The minimum value is 100 records; the maximum value is one million; the default is 500 records. If set to 0, cleanup does not occur.

CORRELATOR

Select trace records with one of the matching correlator identifiers. Up to sixteen(16) identifiers can be specified. Each identifier in the list can be a range: low_number:high_number. Values can be decimal (nnnnnnnnnn) or hexadecimal (X'hhhhhhh'). This filter associates packets in the trace with an IDS event message in syslogd or the system console.

DEBUG(debug_level_list)

Provide documentation about SYSTCPIS format processing. debug_level_list is a list of numbers from 1 to 64. Use only under the direction of an IBM Service representative.

DELAYACK(thresholdl200)

The delay acknowledgement threshold in milliseconds used in the calculation of round-trip time in the TCP session report. The minimum value is 10 milliseconds; the maximum value is 1000 milliseconds; the default value is 200 milliseconds.

DEVTYPE(device_type_list)

Select the type of interface that created the packets. From 1 to 16 types can be specified. The following types can be specified:

- LCSETHERNET
- LCS8023
- LCSETHERNET8023
- LCSTOKENRING
- LCSFDDI
- SNALU62
- HYPERCHANNEL
- CLAWRS6000
- CTC
- CDLC
- ATM
- VIPA
- LOOPBACK
- MPC

- X25
- SNALINK
- IPAQENET
- OSAFDDI
- OSAENET

DHCP[(port_number|67; port_number|68)]

Select DHCP protocol packets. The port_number defines the DHCP port numbers to select trace records for formatting. Equivalent to PORT(67 68).

DNS[(port_number|53)]

Select Domain Name Service protocol packets. The port_number defines the DNS port number to select trace records for formatting. Equivalent to PORT(53).

DOMAIN[(port_number|53)]

Select Domain Name Service protocol packets. The port_number defines the DNS port number to select trace records for formatting. Equivalent to PORT(53).

DUMP[(nnnnnnnn|65535)]

Dump the selected packets in hex with EBCDIC and ASCII translations. The IP and protocol headers are dumped separately from the packet data. The value *nnnnnnnn* is the maximum amount of packet data to be dumped from each packet. The default value is 65535 bytes; the minimum value is 0; the maximum value is 65535. The IP and protocol headers are not subject to this maximum. The default report options are DUMP and FORMAT. The BOTH, ASCII, EBCDIC, and HEX keywords describe how the dumped packets are translated. The default is BOTH.

EBCDIC

Dump trace record data with EBCDIC translation only. The default is BOTH.

EXPORT[(DETAIL|SUMMARY)]

The selected packets are written to the EXPORT data set in .CSV (Comma Separated Value) format. In .CSV format, each character field is surrounded by double quotation marks and successive fields are separated by commas. The file's first line defines the fields. Each subsequent line is a record containing the values for each field.

DETAIL

Format the IP header, protocol header, and protocol data as separate lines of data.

SUMMARY

Format the IP header and protocol header in one line of data. SUMMARY is the default.

Allocate a file with DDNAME of EXPORT before invoking the CTRACE command with EXPORT in the OPTIONS string.

```
ALLOC FILE(EXPORT) DA(PACKET.CSV) SPACE(15 15) TRACK
```

The record format is variable block with logical record length of 512 bytes.

FINGER[(port_number|79)]

Select FINGER protocol packets. The port_number defines the FINGER port number to select trace records for formatting. Equivalent to PORT(79).

FLAGS(flags list)

Select packets that have the matching characteristics. Flags that can be specified are:

ABBREV

Select packets that are abbreviated.

ACK

Select packets that have a TCP header with the ACK flag set.

DF

Select packets that have the don't fragment flag set.

FIC

Select packets that are the first fragment of a packet ($ip_froff = 0$ and $ip_mf = 1$)

FIN

Select packets that have a TCP header with the FIN flag set.

FULL

Select packets that are complete.

HOME

Select packets that have an IP destination address equal to the IP source address.

IN Select packets that are inbound.

IPO

Select packets that have an IP header options field.

LIC

Select packets that are the last fragment of a packet ($ip_froff > 0$ and $ip_mf = 0$).

MIC

Select packets that are the middle fragment of a packet ($ip_froff > 0$ and $ip_mf = 1$).

OIC

Select packets that are not fragmented. ($ip_froff = 0$ and $ip_mf = 0$).

OUT

Select packets that are outbound.

PING

Select packets that are ICMP ECHO or ICMP ECHO REPLY.

PSH

Select packets that have a TCP header with the PSH flag set.

RSM

Select packets that have been reassembled.

RST

Select packets that have a TCP header with the RST flag set.

SEG

Select packets that have been segmented.

SYN

Select packets that have a TCP header with the SYN flag set.

TCPO

Select packets that have a TCP header options field.

TOS

Select packets that have a non zero value in the ip_tos field.

TUNNEL

Select packets that were in a GRE protocol tunnel.

URG

Select packets that have a TCP header with the URG flag set.

ZWIN

Select packets that have a TCP header with a zero window value.

Notes:

1. The use of the FIC, MIC, and LIC flags require the use of the NOREASSEMBLY option.
2. When a packet is reassembled, it becomes an OIC packet with the RSM flag set.

FMT

Equivalent to FORMAT.

FORMAT[(DETAIL|SUMMARY)]

The selected packets with defined packet data are to be formatted. The SHORT keyword on the CTRACE command selects this option if no other report options are specified. The default report options are DUMP and FORMAT.

DETAIL

Format the IP header, protocol header, and the protocol data. This is the default.

SUMMARY

Format the IP header and protocol header. DETAIL is the default.

FTP[(port_number|20; port_number|21)]

Select FTP protocol packets. The port_number defines the FTP port numbers to select trace records for formatting. Equivalent to PORT(20,21).

FULL

Equivalent to DUMP and FORMAT. The FULL keyword on the CTRACE command selects this option if no other report options are specified.

GAIN(rtgain|125,vargain|250)

Used in the calculation of round-trip time in the TCP session report. The time is expressed in milliseconds. The minimum value is 0 milliseconds; the maximum value is 1000 milliseconds.

rtgain The round trip gain value. The default value is 125 milliseconds.

vargain

The variance gain value. The default value is 250 milliseconds.

GOPHER[(port_number|70)]

Select GOPHER protocol packets. The port_number defines the GOPHER port numbers to select trace records for formatting. Equivalent to PORT(70).

GRE

Select trace records with a protocol number of 47. Equivalent to PROTOCOL(47).

GROUP

Select trace records with one of the matching group identifiers. The following group identifiers can be specified:

- TCPTR

- UDPTR
- SCAN
- ATTACK

HEX

Trace record data is not dumped with ASCII or EBCDIC translation. The default is BOTH.

HOST

Select trace records with a host IP address. Equivalent to IPADDR(0.0.0.0/255.255.0.0).

HTTP[(port_number|80)]

Select HTTP protocol packets. The port_number defines the HTTP port numbers to select trace records for formatting. Equivalent to PORT(80). See the WWW keyword.

ICMP

Select trace records with a protocol number of 1. Equivalent to PROTOCOL(1).

IGMP

Select trace records with a protocol number of 2. Equivalent to PROTOCOL(2).

INSTANCE

Select trace records with one of the matching instance identifiers. The identifiers can be in decimal (nnnnn) or hexadecimal (x'hhhhhhh'). The instance identifier is the lower 2 bytes of the PROBEID. Up to 16 identifiers can be specified.

IPADDR(ipaddr[/subnet_mask][X'hhhhhhh'[]-nnnnn.[])

Select packets with a matching IP address, optional subnet mask and optional port number. Up to 16 IP addresses can be specified. The IPADDR is specified in three parts:

1. An IP address

The address can be in dotted decimal notation, a keyword, or a hex value.

- Dotted decimal notation

127.0.0.1

- A keyword

A A class A address, 0.0.0.0/128.0.0.0

B A class B address, 128.0.0.0/192.0.0.0

C A class C address, 192.0.0.0/224.0.0.0

D A class D address, 224.0.0.0/240.0.0.0

E A class E address, 240.0.0.0/248.0.0.0

H A local host address, 0.0.0.0/0.0.255.255

L A loopback address, 127.0.0.0/255.0.0.0

M The broadcast address, 255.255.255.255/255.255.255.255

***** Any address, 0.0.0.0/0.0.0.0

0 An address of zero, 0.0.0.0/255.255.255.255

- Hexadecimal notation

X'7f000001'

2. A submask

The submask is preceded by a slash(/). Specify a submask only when the IP address is in dotted decimal notation. The mask can be in dotted decimal

notation or as a shift value. The subnet shift value is a number less than or equal to 32. Example: 9\8 or 9.37\16

3. A port number

The port number is preceded by a dash (—). It is a decimal number in the range 0–65535.

Notes:

1. There should be no spaces between the IP addresses and the subnet masks.
2. The BROADCAST, CLASSA, CLASSB, CLASSC, CLASSD, CLASSE, HOST, and LOOPBACK keywords add to the total of 16 IP addresses.
3. The port number, when used, adds to the the total of 16 port numbers in the PORT keyword.

IPID(ipid_number_list)

Select packets that match the ip_id number in the IP packet header. Up to 16 ID numbers can be specified in the range 0–65535. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnnnn) or hexadecimal (x'hhhh'). If the packets have been fragmented, specify NOREASSEMBLY to select each packet.

JOBLISTJOBNAME(job_name_list)

Select data trace records with the specified JOBNAME. Up to 16 job names can be specified. Each job name can be up to 8 characters in length. If the last character of a job name is an asterisk (*) then only the characters up to the asterisk are compared.

The CTRACE JOBLIST/JOBNAME parameter provides the same function, except that wildcards are not supported.

LINKNAME(link_name_list)

Select packet trace records with the specified LINKNAME. Up to 16 link names can be specified. Each link name can be up to 16 characters in length. If the last character of a link name is an asterisk (*) then only the characters up to the asterisk are compared.

LOOPBACK

Select trace records with a loop back address. Equivalent to IPADDR(127.0.0.0/255.0.0.0).

NOREASSEMBLY

Do not reassemble fragmented IP packets into a complete packet. This is the default.

NOSEGMENT

Packet trace records that span multiple Ctrace records are not recombined. Only the first segment record of a packet is used. The rest of the segment records are discarded. SEGMENT is the default.

NOT

If the NOT option is selected then any selection criteria is reversed. If a record matches the selection criteria, it is not processed. If a record does not match the selection criteria, it is processed. If no selection criteria were found in the OPTIONS(()) keyword then the NOT option has no effect.

OPTION

The selected options with defaults are listed.

OSPF

Select packets with a protocol number of 89. Equivalent to PROTOCOL(89).

PORT(port_number_list)

Select trace records with one of the specified port numbers. Up to 16 port numbers can be specified in the range 0–65535.

Note: The BOOTP, DHCP, DNS, DOMAIN, FINGER, GOPHER, HTTP, RIP, ROUTER, RPC, SMTP, SNMP, TELNET, TFTP, TIME, and WWW keywords add to the total number of ports.

PROBEID

Select trace records with one of the matching probe identifiers. The identifiers can be expressed in decimal (nnnnn) or hexadecimal (x'hhhhhhh'). Up to 16 identifiers may be specified.

Note: The probe identifiers can also be specified on the ENTIDLIST keyword of the CTRACE subcommand.

Refer to the *z/OS Communications Server: IP and SNA Codes* for additional information about probe identifiers.

PROTOCOL(protocol number list)

Select trace records with one of the specified protocol numbers. Up to 16 protocol numbers can be specified in the range 0–255. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnn) or hexadecimal (X'hh').

Note: The ICMP, IGMP, OSPFI, SESSION, TRANSIT, TCP, and UDP keywords add to the total number of protocols.

QOS(quality_of_service_list)

Select the records with the matching quality of service from the ip_tos field. Up to 16 QOS values can be specified in the range 0–7. Each entry in the list can be a range: low_number:high_number. Values can be decimal (n) or hexadecimal (X'h').

REASSEMBLY[(packet_size|65535)]

Reassemble IP fragments into a complete packet. **packet_size** is the maximum size of a reassembled packet that is allowed. The smallest value allowed is 576 bytes, the largest is 65535 bytes. The default value is 65535 bytes. NOREASSEMBLY is the default.

RECORDS(record_number_list)

Select the records with the matching record numbers in the trace data. Up to sixteen (16) record numbers can be specified. Record numbers are assigned after any IPCS CTRACE selection criteria have been met. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnnnnnnnnn) or hexadecimal (X'hhhhhhh').

RIP[(port_number|520)]

Select RIP protocol packets. The **port_number** defines the RIP port number to select trace records for formatting. Equivalent to PORT(520).

ROUTER[(port_number|520)]

Select RIP protocol packets. The **port_number** defines the RIP port number to select trace records for formatting. Equivalent to PORT(520).

RPC[(port_number|111)]

Select RPC protocol packets. The **port_number** defines the RPC port number to select trace records for formatting. Equivalent to PORT(111).

SEGMENT

Packet trace records that span multiple Ctrace records are recombined. Data

from segment records is saved until all the Ctrace records have been read to recreate the original packet. SEGMENT is the default.

SESSION[(DETAIL|STATE|SUMMARY)]

Generate a report showing TCP or UDP session traffic.

DETAIL

List each of the packets for a session, and the summary statistics.
DETAIL is the default.

STATE

List the beginning and ending state of each session.

SUMMARY

Show only the summary statistics.

Note: The UDP session analysis is also used for other protocols.

SMTP[(port_number|25)]

Select SMTP protocol packets. The **port_number** defines the SMTP port number to select trace records for formatting. Equivalent to PORT(25).

SNIFFER[(TOKENRING|ETHERNET)]

Formats the trace records so that they are suitable for downloading to a DatagLANce* Network Analyzer or a Sniffer** Network Analyzer for analysis. The trace records are written to the file with a DD name of SNIFFER. Once the file is generated, it can be downloaded as a binary file to the analyzer and loaded using the standard features of the analyzer. Use NOREASSEMBLY to prevent the formatter from reassembling packets. Each packet will be passed as the packets were collected. The logical record length of the SNIFFER data set determines the largest amount of packet data written to the data set.

TOKENRING

If this keyword is specified, the output is formatted for the token-ring analysis application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on a token ring will be displayed. Packets from all devices can be displayed using this option. If you are using the token-ring analyzer application, the DOS file extension must be TRC.

ETHERNET

If this keyword is specified, the output is formatted for the Ethernet analysis application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on an Ethernet will be displayed. Packets from all devices can be displayed using this option. If you are using the Ethernet analyzer application, the DOS file extension must be ENC.

The default for the SNIFFER option is TOKENRING.

SNMP[(port_number|161 port_number|162)]

Select SNMP protocol packets. The **port_number** defines the SNMP port number to select trace records for formatting. Equivalent to PORT(161 162).

SPEED[(local|10,remote|10)]

The link speed, in megabits per second, for the local and remote link. These values are used in throughput calculations in the TCP session report. Valid values are in the range 0–18010. The default value is 10. Specify the slowest speed of the link in the route.

STATISTICS[(DETAIL|SUMMARY)]

After all the records have been processed, generate a report showing the number of records selected by record type, Device type, Jobname, Linkname,

Protocol number, IP address, and port numbers. TALLY on the CTRACE command selects this option if no other report options are specified.

STATS

Equivalent to the STATISTICS option.

STREAMS[(nnn|128)]

Collect the packet data for dumping or formatting after the trace file has been processed. **nnn** is the maximum amount of storage used to capture each stream. The smallest value is 16KB. The largest value is 512KB. The default value is 128KB. The value is in 1024 bytes (1K) units.

SUMMARY

Format a single line for each trace record. SUMMARY on the CTRACE command selects this option if no other report options are specified.

TALLY

Equivalent to the STATISTICS option.

TCP

Select trace records with a protocol number of 6. Equivalent to PROTOCOL(6).

TELNET[(port_number|23)]

Select TELNET protocol packets. The **port_number** defines the TELNET port number to select trace records for formatting. Equivalent to PORT(23).

TFTP[(port_number|69)]

Select TFTP protocol packets. The **port_number** defines the TFTP port number to select trace records for formatting. Equivalent to PORT(69).

TIME[(port_number|37)]

Select TIME protocol packets. The **port_number** defines the TIME port number to select trace records for formatting. Equivalent to PORT(37).

TOD

Use the time the trace data was captured for the reports. Normally the time the trace data was moved to the trace buffer is shown.

Note: The CTRACE command uses the time stamp when the trace data was moved to the buffers for START and STOP time selection.

TYPE(probe type identifier)

Select trace records with one of the matching probe type identifiers. The **probe type identifier** is the second byte of the probe identifier. Up to 16 identifiers can be specified. The following probe types can be used:

Identifier	Type
------------	------

0100	
------	--

TCPTR	
-------	--

0200	
------	--

UDPTR	
-------	--

0301	
------	--

VSSCAN	
--------	--

0303	
------	--

NORMSCAN	
----------	--

0401	
------	--

MALFORMED	
-----------	--

0402
RAW

0403
IPFRAGMENT

0404
ICMP

0405
IPOPT

0406
IPPROTO

0407
FLOOD

0408
PREPECHO

UDP

Select trace records with a protocol number of 17. Equivalent to PROTOCOL(17).

USEREXIT(exitname)

Names the user exit to be called for each selected record. The USEREXIT is called after the record passes the other filter options. It is passed the same parameter list as the CTRACE user exit. A nonzero return code indicates the record is not selected for formatting. The USEREXIT keyword on the CTRACE command names a user exit that is called before the SYSTCPIS trace record filtering is done. If this exit routine returns a nonzero return code, then the record is skipped by the SYSTCPIS formatter.

WWW[(port_number|80)]

Select HTTP protocol packets. The **port_number** defines the HTTP port number to select trace records for formatting. Equivalent to PORT(80).

IDS reports

The SYSTCPIS Ctrace formatter is based on the SYSTCPDA formatter (and in fact shares many of the data structures and format routines) and includes the reports for the SYSTCPDA formatter. However, the REASSEMBLY, SESSION and STREAMS reports may prove of little value for the SYSTCPIS, since they depend on having a more complete set of packets.

- The STATISTICS report (both SUMMARY and DETAIL) provide an overview of the data collected.
- The SUMMARY report provides one line per IDS event.
- The FORMAT, and DUMP reports will format individual packets.
- The EXPORT and SNIFFER options will write the packet to an external file for later analysis.

Reports

The following sections describe the various reports available.

OPTION:

Purpose: List the selected options and default keyword values.

Format: The following command was used to obtain the example of this report.

```
CTRACE COMP(SYSTCPIS) DSN('D74L.PQ33208.PTRACE3')+  
OPTION((OPT SESS FORM))  
Report
```

Examples:

```
COMPONENT TRACE SHORT FORMAT  
SYSNAME(MVST)  
COMP(SYSTCPIS)  
OPTIONS((OPT SESS FORM))  
1 DSNNAME('D74L.PQ33208.PTRACE3')  
2 OPTIONS((  
  Session Format Option  
  Skip(1) Limit(999999999) Dump(65535) Both Interval(5) MaxRsm(32767)  
  Bootp(67,68) Domain(53) Finger(79) Ftp(20,21)  
  Gopher(70) Router(520) Rpc(111) SMTP(25)  
  SNMP(161,162) Telnet(23) Tftp(69) Time(37) WWW(80)  
  User( )  
  Protocol( /* 1 */  
            6 /* TCP */  
  )  
))
```

The following describes numbered areas of the example.

1

DSNAME is the name of the source data.

2

OPTIONS(...) is a listing of the active options with default values.

SUMMARY:

Purpose: Show one line of information about each record in the trace.

Format: The following command was used to obtain the example of this report.

```
CTRACE COMP(SYSTCPIS) SUMMARY DSN('D74L.PQ33208.PTRACE3')
```

Examples:

```
SYSNAME(MVST)
COMP(SYSTCPDA)
DSNAME('D74L.PQ33208.PTRACE3')
< 1 21:06:23.817394 ATTACK 00000001 FFFFFFFF 192.168.200.1-520
255.255.255.255-520 UDP
< 2 21:06:23.817414 ATTACK 00000001 FFFFFFFF 192.168.200.1-520
255.255.255.255-520 UDP
< 3 21:06:23.817426 ATTACK 00000001 FFFFFFFF 192.168.200.1-520
255.255.255.255-520 UDP
< 4 21:06:23.821040 ATTACK 00000001 FFFFFFFF 192.168.200.1-520
255.255.255.255-520 UDP
< 5 21:06:23.821051 ATTACK 00000001 FFFFFFFF 192.168.200.1-520
255.255.255.255-520 UDP
< 6 21:06:23.821213 ATTACK 00000001 FFFFFFFF 192.168.201.1-520
255.255.255.255-520 UDP
.
< 78 21:14:21.743062 SCAN 00000001 C0A8C805 192.168.58.11-53
192.168.200.5-1030 TCP
< 79 21:14:21.745910 SCAN 00000001 C0A8C805 192.168.58.11-53
192.168.200.5-1030 TCP
> 80 21:14:21.745995 SCAN 00000001 C0A83A0B 192.168.200.5-1030
192.168.58.11-53 TCP
=====
=====
Packet Trace Statistics
80 Ctrace records processed
1 segmented trace records read
1 segmented trace records were lost
79 trace records read
0 records could not be validated
79 records passed filtering
79 packet trace records processed
0 data trace records processeed
```

The following describe areas of the example.

I Inbound packet.

O Outbound packet.

nnnnn

The Ctrace record number.

hh:mm:ss.mmmmmmm

The time stamp of the record.

ATTACK

The group assigned to the trace record.

PROBEID

The probe identifier assigned to the trace record.

nnn.nnn.nnn.nnn-ppppp

The source IP address and port number.

nnn.nnn.nnn.nnn-ppppp

The destination IP address and port number.

Protocol

The protocol in the IP header.

Type	Hexadecimal Value	Decimal value
TCP	6	6
UDP	11	17
ICMP	1	1
ICMPV6	3A	58
IGMP	2	2
GRE	2F	47
ESP	32	50

FORMAT:

Purpose: Format the Ctrace record header, the IP packet header, the protocol header, and the packet data. If one of the ports is a well-known port number and the SYSTCPIS supports data for the port number, the packet data is shown.

Format: The following command was used to obtain the example of this report.

```
CTRACE COMP(SYSTCPIS) DSN('D74L.PQ33208.PTRACE3') +  
OPTION((FORMAT))
```

Examples:

```
COMPONENT TRACE SHORT FORMAT  
SYSNAME(MVST)  
COMP(SYSTCPIS)  
OPTIONS((OPT))  
DSNAME('D74L.PQ33208.PTRACE3')  
OPTIONS((  
  Format Option  
  Skip(1) Limit(999999999) Dump(65535) Both Interval(5) MaxRsm(32767)  
  Bootp(67,68) Domain(53) Finger(79) Ftp(20,21)  
  Gopher(70) Router(520) Rpc(111) SMTP(25)  
  SNMP(161,162) Telnet(23) Tftp(69) Time(37) WWW(80)  
  User(      )  
))  
1 1999/11/11  
2 2 MVST ATTACK 00000001 21:06:23.817414 Attack Trace Test  
3 From Link : LMVST0 Device: CLAW_RS/6000 Full=532  
Tod Clock : 1999/11/11 21:06:23.817408 Module: EZBCTINI  
Job Name : **N/A*** Asid: 0000 Tcb: 45C0220  
Cid : C0A8C801 Instance: 00000000  
Event Id : 00000001 Correlator: FFFFFFFF  
Policy : SYSTCPDA  
4 IpHeader: Version : 4 Header Length: 20  
Precedence : Internetwork control TOS:  
Packet Size : 532 ID Number: 0  
Fragment : Offset: 0  
TTL : 2 Protocol: UDP CheckSum: 2D70 FFFF  
Source : 192.168.200.1  
Destination : 255.255.255.255 Broadcast  
5 UDP  
Source Port : 520 (router) Destination Port: 520 (router)  
Datagram Length : 512 CheckSum: 0000 FFFF  
6 -- RIP Packet Received ----- Type: Reply (V1)  
Destination_Addr : 192.168.200.0 Metric: 1 Family: 2  
Destination_Addr : 192.168.200.2 Metric: 1 Family: 2  
Destination_Addr : 192.168.200.3 Metric: 1 Family: 2  
Destination_Addr : 192.168.200.4 Metric: 1 Family: 2  
Destination_Addr : 192.168.200.5 Metric: 1 Family: 2  
Destination_Addr : 0.0.0.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.104.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.105.0 Metric: 8 Family: 2  
Destination_Addr : 192.200.10.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.107.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.110.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.96.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.98.0 Metric: 2 Family: 2  
Destination_Addr : 192.168.99.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.101.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.102.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.103.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.72.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.73.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.77.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.78.0 Metric: 8 Family: 2  
Destination_Addr : 192.168.79.0 Metric: 8 Family: 2
```

Destination_Addr : 192.168.64.0	Metric: 8	Family: 2
Destination_Addr : 192.168.65.0	Metric: 8	Family: 2
Destination_Addr : 192.168.66.0	Metric: 8	Family: 2

25 Rip entries found

The following describes numbered areas of the example.

1

The date of the trace records.

2

A summary line indicating the source of the trace record showing:

- The record number.
- The system name.
- The group name.
- The probe ID value (in hex).
- The time the record was moved to the trace buffer, or with the TOD option the time the trace data was captured.
- The description of the IDS event associated with the probe.

3

The trace header with these fields:

- The direction of the trace record: *From* or *To*.
- The link name.
- The device type.
- *Full* or *Abbrev* with amount of trace data available.
- The time the trace record was captured.
- The job name associated when the probe was triggered.
- The asid of the address space when the probe was triggered.
- The system tcb pointer when the probe was triggered (or zero if in SRB mode).
- The CID (communications ID) of the session.
- The instance identifier, the lower 2 bytes of the PROBEID.
- The Event identifier, the upper 2 bytes of the PROBEID.
- The Correlator identifier.
- The name of the current policy. This may be the policy that triggered the probe or the name of the policy the session was using at the time the probe was triggered.

4

The IP header showing fields from the IP version 4 header. The header length is the number bytes for the header. The offset field is the number bytes from the end of the IP header of where the fragment appears. With the REASSEMBLY option active, this field always displays zeros.

5

The UDP header.

6

Depending on the port number the trace data is formatted. In this case it is a RIP packet.

7

The TCP protocol header. The fields of the header are shown. The sequence numbers and window size are shown in decimal.

8

The length of the DNS packet data following is shown.

9

The DNS header and resource records are formatted.

Note: If possible the check sum of the packet is calculated. If the calculated value is X'FFFF' then the check sum is correct. If X'0000' the check sum could not be calculated. The packet was incomplete or fragmented. Other values indicate a check sum error.

Using the protocol numbers and the well known port numbers, format routines are invoked to format standard packet data records. The port number for the PORT keywords define the port numbers to be used to invoke a format routine.

Port
Keyword

67, 68
BOOTP

67, 68
DHCP

53 Domain

79 Finger

70 Gopher

520
Rip

520
Router

111
RFC

25 SMTP

23 TELNET

69 TFTP

37 TIME

DUMP:

Purpose: Format the IP header, protocol header and packet data in hex. The data can also be translated into EBCDIC, ASCII or both.

Format: The following command was used to obtain the example of this report.

```
CTRACE COMP(SYSTCPIS) DSN('D74L.PQ33208.PTRACE3')+  
OPTION((OPT DUMP))
```

Examples:

```
COMPONENT TRACE SHORT FORMAT  
SYSNAME(MVST)  
COMP(SYSTCPIS)  
OPTIONS((OPT DUMP))  
DSNAME('D74L.PQ33208.PTRACE3')  
OPTIONS(  
  Dump Option  
  Skip(1) Limit(999999999) Dump(65535) Both Interval(5) MaxRsm(32767)  
  Bootp(67,68) Domain(53) Finger(79) Ftp(20,21)  
  Gopher(70) Router(520) Rpc(111) SMTP(25)  
  SNMP(161,162) Telnet(23) Tftp(69) Time(37) WWW(80)  
  User( )  
)  
1999/11/11  
1 MVST SYSTCPIS SYSTCPIS 21:06:23.817394 576 Packet Trace  
1 IP Header : 20  
000000 45C00214 00000000 02112D70 C0A8C801 FFFFFFFF  
2 Protocol : 8  
000000 02080208 02000000  
3 Data : 504  
000000 02010000 00020000 C0A8C800 00000000 00000000 00000001 00020000 C0A8C802  
|.....yH.....yH.....|  
000020 00000000 00000000 00000001 00020000 C0A8C803 00000000 00000000 00000001  
|.....yH.....|  
000040 00020000 C0A8C804 00000000 00000000 00000001 00020000 C0A8C805 00000000  
|....yH.....yH.....|  
000060 00000000 00000001 00020000 00000000 00000000 00000000 00000008 00020000  
|.....|  
000080 C0A86800 00000000 00000000 00000008 00020000 C0A86900 00000000 00000000  
|.y.....y.....h.....i.....|  
0000A0 00000008 00020000 C0C80A00 00000000 00000000 00000008 00020000 C0A86B00  
|.....H.....y,. .....k.|  
0000C0 00000000 00000000 00000008 00020000 C0A86E00 00000000 00000000 00000008  
|.....y>.....n.....|  
0000E0 00020000 C0A86000 00000000 00000000 00000008 00020000 C0A86200 00000000  
|....y-.....y.....b.....|  
000100 00000000 00000002 00020000 C0A86300 00000000 00000000 00000008 00020000  
|.....y.....c.....|  
000120 C0A86500 00000000 00000000 00000008 00020000 C0A86600 00000000 00000000  
|.y.....y.....e.....f.....|  
000140 00000008 00020000 C0A86700 00000000 00000000 00000008 00020000 C0A84800  
|.....y.....y.....g.....H.....|  
000160 00000000 00000000 00000008 00020000 C0A84900 00000000 00000000 00000008  
|.....y.....I.....|  
000180 00020000 C0A84D00 00000000 00000000 00000008 00020000 C0A84E00 00000000  
|....y(.....y+.....M.....N.....|  
0001A0 00000000 00000008 00020000 C0A84F00 00000000 00000000 00000008 00020000  
|.....y|.....O.....|  
0001C0 C0A84000 00000000 00000000 00000008 00020000 C0A84100 00000000 00000000  
|.y.....y.....@.....A.....|  
0001E0 00000008 00020000 C0A84200 00000000 00000000 00000008  
|.....y.....B.....|
```

The following describes numbered areas of the example.

- 1** The IP header is dumped with no translation.
- 2** The protocol header is dumped with no translation.
- 3** The packet data is dumped with the translation specified by the ASCII, BOTH, EBCDIC, or HEX keyword. The default is BOTH. The amount of data dumped can be limited by the value specified with the DUMP keyword. The default is 65535 bytes.

SNIFFER:

Purpose: This report shows information written to the SNIFFER data set.

Format: The following command was used to obtain the example of this report.

```
ALLOC F(SNIFFER) DATASET(SNIFFER.TRC) LRECL(1600) RECFM(V B) +  
REUSE TRACK SPACE(15 15)
```

```
CTRACE COMP(SYSTCPIS) DSN('MWS.PQ33208.PTRACE4')+  
OPTION((OPT TALLY SNIFFER NOREASSEMBLY))
```

Examples:

```
COMPONENT TRACE SHORT FORMAT  
SYSNAME(MVS142)  
COMP(SYSTCPIS)  
OPTIONS(( OPT TALLY SNIFFER NOREASSEMBLY))  
DSNAME('MWS.PQ33208.PTRACE4')  
PTRPT04I SNIFFER(TOKENRING) option selected
```

```
OPTIONS((  
  Statistics PacketTrace X25 Sniffer Option Tokenring  
  Skip(0) Limit(999999999) Dump(65535) Both Interval(5) MaxRsm(32767)  
  Stream(131072)  
  Bootp(67,68) Domain(53) Finger(79) Ftp(20,21)  
  Gopher(70) Router(520) Rpc(111) SMTP(25)  
  SNMP(161,162) Telnet(23) Tftp(69) Time(37) WWW(80)  
  User(  
  )  
))  
  108 records written to SNIFFER  
46,000 bytes written to SNIFFER  
  22 records were truncated to 1600 bytes
```

The following describes areas of the example.

108 records written to SNIFFER

This record count includes the two header records and one trailer record that were written to the SNIFFER data set.

46,000 bytes written to SNIFFER

The number of data bytes written to the SNIFFER data set. This should be close to the amount of data to be downloaded.

22 records were truncated to 1600 bytes

Because the logical record length was 1,600 bytes, 22 records were truncated. This can be avoided by increasing the logical record length. The maximum logical record length is 32,763 or the size of physical disk blocks, whichever is smaller.

STATISTICS:

Purpose: The records are counted by record type (probeID), device type, link name, job name, Asid, protocol, IP address, TCP port number, UDP port number, Correlator identifier, Event identifier, Group identifier, and Instance identifier.

Format: The following command was used to obtain the example of this report.

```
CTRACE COMP(SYSTCPIS) DSN('D74L.PQ33208.PTRACE3')+  
OPTION((OPT STATS))
```

Examples:

```
COMPONENT TRACE SHORT FORMAT  
SYSNAME(MVST)  
COMP(SYSTCPIS) SUBNAME((TCPIP))  
OPTIONS((OPT STATIST))  
DSNAME('D74L.PQ33208.PTRACE3')  
OPTIONS(  
  Statistics PacketTrace X25 DataTrace Reassembly Option  
  Skip(1) Limit(999999999) Dump(65535) Both Interval(5) MaxRsm(32767)  
  Bootp(67,68) Domain(53) Finger(79) Ftp(20,21)  
  Gopher(70) Router(520) Rpc(111) SMTP(25)  
  SNMP(161,162) Telnet(23) Tftp(69) Time(37) WWW(80)  
  User( )  
)
```

No packets needed reassembly

1 Packet Trace Statistics

```
80 Ctrace records processed  
2 segmented trace records read  
0 segmented trace records were lost  
79 trace records read  
0 records could not be validated  
79 records passed filtering  
79 packet trace records processed  
0 data trace records processed
```

2 Record Type Report

Record Type	Total	Input	Data	Output	Data First	yy/mm/dd hh.mm.ss
Last yyyy/mm/dd hh.mm.ss						
Packet Trace	79	54	70651	25	4353	1 1999/11/11 21:06:23
80 1999/11/11 21:14:21						
Total:	79	54	70651	25	4353	

1 Record Type(s) found

3 Device Type Report

Device Type	Total	Input	Data	Output	Data First	yy/mm/dd hh.mm.ss
Last yyyy/mm/dd hh.mm.ss						
21(CLAW_RS/6000)	73	48	70414	25	4353	1 1999/11/11 21:06:23
80 1999/11/11 21:14:21						
33(VIPA)	6	6	237	0	0	27 1999/11/11 21:07:04
38 1999/11/11 21:07:13						
Total:	79	54	70651	25	4353	

2 Device Type(s) found

4 Link Name Report

Link Name	Total	Input	Data	Output	Data First	yy/mm/dd hh.mm.ss
Last yyyy/mm/dd hh.mm.ss						
LMVST0	48	33	63814	15	193	1 1999/11/11 21:06:23
80 1999/11/11 21:14:21						
LMVST1	25	15	6600	10	4160	6 1999/11/11 21:06:23
48 1999/11/11 21:07:21						
VLMVST	6	6	237	0	0	27 1999/11/11 21:07:04
38 1999/11/11 21:07:13						
Total:	79	54	70651	25	4353	

3 Link Name(s) found

5 Protocol Report

Protocol	Total	Input	Data	Output	Data First	yy/mm/dd hh.mm.ss
----------	-------	-------	------	--------	------------	-------------------

```

Last yyyy/mm/dd hh.mm.ss
1(ICMP) 3 3 108 0 0 28 1999/11/11 21:07:04
38 1999/11/11 21:07:13
6(TCP) 27 16 56502 11 60 53 1999/11/11 21:14:21
80 1999/11/11 21:14:21
17(UDP) 49 35 14041 14 4293 1 1999/11/11 21:06:23
52 1999/11/11 21:14:16
Total: 79 54 70651 25 4353
3 Protocol(s) found
6 IP Address Report
IP Address Total Input Data Output Data First yyyy/mm/dd hh.mm.ss
Last yyyy/mm/dd hh.mm.ss
192.168.58.11 31 18 57214 13 105 49 1999/11/11 21:14:16
80 1999/11/11 21:14:21
192.168.98.2 6 6 237 0 0 27 1999/11/11 21:07:04
38 1999/11/11 21:07:13 192.168.200.1 17 15 6600 2 88
1 1999/11/11 21:06:23
43 1999/11/11 21:07:21
192.168.200.5 33 18 57214 15 193 11 1999/11/11 21:06:38
80 1999/11/11 21:14:21
192.168.201.1 25 15 6600 10 4160 6 1999/11/11 21:06:23
48 1999/11/11 21:07:21
192.168.201.5 10 0 0 10 4160 12 1999/11/11 21:06:38
36 1999/11/11 21:07:08
255.255.255.255 30 30 13200 0 0 1 1999/11/11 21:06:23
48 1999/11/11 21:07:21
Total: 152 102 141065 50 8706
7 IP Address(s) found
7 Tcp Port Report
Tcp Port Total Input Data Output Data First yyyy/mm/dd hh.mm.ss
Last yyyy/mm/dd hh.mm.ss
53(domain) 27 16 56502 11 60 53 1999/11/11 21:14:21
80 1999/11/11 21:14:21
1030() 27 16 56502 11 60 53 1999/11/11 21:14:21
80 1999/11/11 21:14:21
Total: 54 32 113004 22 120
2 Tcp Port(s) found
8 Udp Port Report
Udp Port Total Input Data Output Data First yyyy/mm/dd hh.mm.ss
Last yyyy/mm/dd hh.mm.ss
53(domain) 4 2 712 2 45 49 1999/11/11 21:14:16
52 1999/11/11 21:14:16
161(snmp) 3 3 129 0 0 27 1999/11/11 21:07:04
37 1999/11/11 21:07:13
520(router) 42 30 13200 12 4248 1 1999/11/11 21:06:23
48 1999/11/11 21:07:21
4319() 3 3 129 0 0 27 1999/11/11 21:07:04
37 1999/11/11 21:07:13
Total: 52 38 14170 14 4293
4 Udp Port(s) found
9 79 records processed for this report
4K of storage used to create this report
22 requests for 1584 bytes of storage were made

```

The following describes numbered areas of the example.

1

The standard statistics shown with all executions of the SYSTCPIS packet trace formatter.

- **80 Ctrace records processed**

The total number of Ctrace records given to the SYSTCPIS packet trace formatter.

- **2 segmented trace records read**

The total number of packets that spanned multiple Ctrace records.

- **0 segmented trace records were lost**
The total number of packets records that could not be put back together.
- **79 trace records read**
The total number of complete trace records.
- **0 records could not be validated**
The number of incomplete Ctrace records that could not be used.
- **79 records passed filtering**
The number of records that were successfully formatted.
- **79 packet trace records processed**
The number of records that were packet trace records.
- **0 data trace records processeed**
The number of records that were data trace records.

2

The totals by record type (ProbeID).

3

The totals by device type for packet trace records.

4

The totals by Link Name for packet trace records.

5

The totals by Protocol number for packet trace records.

6

The totals by IP Address. Both the destination and source IP addresses are counted except when they are the same in a record.

7

The totals by TCP Port number. Both the destination and source port numbers are counted except when they are the same in a record.

8

The totals by UDP port number. Both the destination and source port numbers are counted except when they are the same in a record.

9

The number of records processed for the statistics report, the number of 1KB blocks of storage required for this report, the number of requests for storage, and the total amount of storage required for the requests. Not shown is the report by Jobname for data trace records. Each category of totals is broken down by:

- The total number of records.
- The total number of inbound records.
- The total amount of inbound data.
- The total number of outbound records.
- The total amount of outbound data.
- The record number of the first record.
- The time stamp of the first record.
- The record number of the last record.
- The time stamp of the last record.

STREAM:

Purpose: There are times where messages span multiple packets. TELNET and DNS are examples. The STREAM report (with the DUMP or FORMAT keywords) capture the entire stream of data.

Format: The following command was used to obtain the example of this report.

```
CTTRACE COMP(SYSTCPIS) DSN(CTRACE.DATA)+  
OPTION((OPT STREAM DUMP ASCII))
```

OMPROUTE trace (SYSTCPRT)

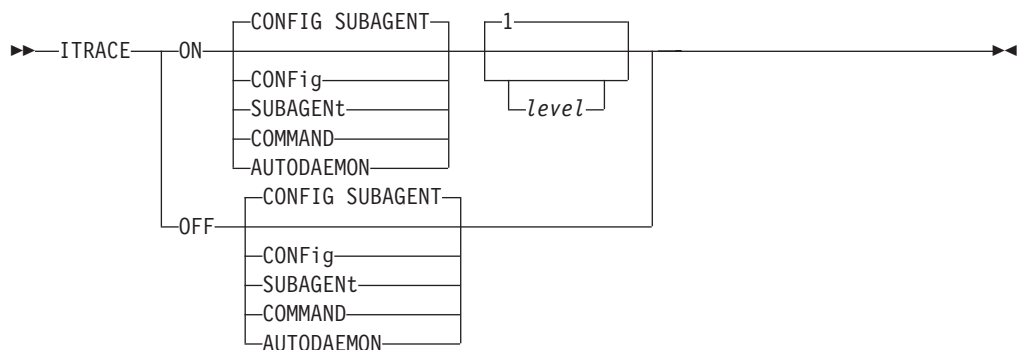
TCP/IP Services component trace is also available for use with the OMPROUTE application. See “TCP/IP services component trace for OMPROUTE” on page 615.

RESOLVER trace (SYSTCPRE)

TCP/IP Services component trace is also available for use with the RESOLVER application. See Chapter 32, “Diagnosing resolver problems” on page 683.

Configuration profile trace

You can use the ITRACE statement in the PROFILE.TCPIP data set to activate TCP/IP run-time tracing for configuration, the TCP/IP SNMP subagent, commands, and the autolog subtask. ITRACE should only be set at the direction of an IBM Support Center representative.



Following are descriptions of the ITRACE parameters:

ON

Select ON to establish run-time tracing. ITRACE ON commands are cumulative until an ITRACE OFF is issued.

OFF

Select OFF to terminate run-time tracing.

CONFig

Turn internal trace for configuration ON or OFF.

SUBAGent

Turn internal trace for TCP/IP SNMP subagent ON or OFF.

COMMAND

Turn internal trace for command ON or OFF.

AUTODAEMON

Turn internal trace for the autolog subtask ON or OFF.

level

Indicates the tracing level to be established. Levels are as follows:

Levels for CONFIG

- 1** ITRACE for all of config
- 2** General level of tracing for all of config
- 3** Tracing for configuration set commands

- 4** Tracing for configuration get commands
- 5** Tracing for syslog calls issued by config
- 100** Tracing for the parser
- 200** Tracing for scanner
- 300** Tracing for mainloop
- 400** Tracing for commands

Levels for SUBAGENT

- 1** General subagent tracing
- 2** General subagent tracing, plus DPI® traces
- 3** General subagent tracing, plus extended storage dump traces
- 4** All trace levels

Level for COMMAND

- 1** ITRACE for all commands

Following is an example illustrating how to use the ITRACE command:

```
ITRACE ON CONFIG 3
ITRACE OFF SUBAGENT
```

Trace output is sent to the following locations:

- Subagent trace output is directed to the syslog daemon. This daemon is configured by the /etc/syslog.conf file and must be active.
- AUTOLOG trace output goes to ALGPRINT.
- Trace output for other components goes to SYSPRINT.

Chapter 6. IPCS subcommands for TCP/IP

The IPCS subcommands for TCP/IP are used to format data from IPCS system dumps. This chapter describes the subcommands (including description, syntax, parameters, and sample output), installation, entering, and execution.

There are two types of subcommands.

- Many of the TCP/IP subcommands work on a specific stack. These subcommands are grouped under the TCPIPICS subcommand to share the TCP (to select the stack) and TITLE options.
- The remaining TCP/IP IPCS subcommands do not require a TCP/IP stack, so they are not under the TCPIPICS subcommand.

Table 13 lists all the IPCS subcommands.

Note: The TCP/IP IPCS commands are not supported for IPCS "active."

Table 13. TCP/IP IPCS commands - showing the TCPIPICS commands first, followed by the general commands.

Command	Description	Alias
TCPIPICS ALL	Equivalent to TCPIPICS STATE TSEB TSDB TSDX DUAF CONFIG ROUTE SOCKET STREAM RAW TCB UDP LOCK TIMER STORAGE	
TCPIPICS API	Display control blocks for Sockets Extended Assembler Macro and Pascal APIs	
TCPIPICS CONFIG	Display device configuration information	TCPIPICS CNFG TCPIPICS CONF
TCPIPICS CONNECTION	Display active or all connections	TCPIPICS CONN
TCPIPICS COUNTERS	Display information about TCP/IP internal execution statistics	
TCPIPICS DETAIL	Equivalent to TCPIPICS TSEB TSDB TSDX DUAF	TCPIPICS CBS
TCPIPICS DU	Equivalent to TCPIPICS DUAF DUCB	
TCPIPICS DUAF	Summarize DUCBs	TCPIPICS DUCBS
TCPIPICS DUCB	Find and format DUCBs	
TCPIPICS FIREWALL	Display information about Firewall filters and tunnels	
TCPIPICS FRCA	Display state information about FRCA connections and objects	
TCPIPICS HASH	Display TCP/IP data stored in hash tables	
TCPIPICS HEADER	Display dump Header info	TCPIPICS HDR
TCPIPICS HELP	Display syntax help for TCPIPICS command	TCPIPICS ?
TCPIPICS LOCK	Display locks	TCPIPICS LOCKSUM
TCPIPICS MAP	Display storage map	

Table 13. TCP/IP IPCS commands - showing the TCPIP commands first, followed by the general commands. (continued)

Command	Description	Alias
TCPIP MTABLE	Display module table	
TCPIP POLICY	Display service policy data	
TCPIP PROFILE	Display TCP/IP configuration data in the format of a profile dataset	TCPIP PROF
TCPIP PROTOCOL	Invokes RAW, TCB, UDP	
TCPIP RAW	Display RAW control blocks	TCPIP MRCB TCPIP RAWSUM TCPIP RCB
TCPIP ROUTE	Display routing information	TCPIP RTE
TCPIP SOCKET	Display socket information	TCPIP SCB TCPIP SOCKSUM
TCPIP STATE	Display general stack information	TCPIP
TCPIP STORAGE	Display TCP/IP storage usage	TCPIP STOR
TCPIP STREAM	Display streams information	TCPIP SKSH TCPIP STREAMS
TCPIP SUMMARY	Equivalent to TCPIP DUAF CONFIG SOCKET	
TCPIP TCB	Display TCP protocol control blocks	TCPIP MTCB TCPIP TCBSUM
TCPIP TELNET	Display Telnet information	
TCPIP TIMER	Display information about timers	TCPIP TIMESUM
TCPIP TRACE	Display TCP/IP CTrace information	TCPIP TCA
TCPIP TREE	Display information about data stored in Patricia trees	TCPIP TREESUM
TCPIP TSDB	Format TSDB	
TCPIP TSDX	Format TSDX	
TCPIP TSEB	Format TSEB	
TCPIP UDP	Display UDP control blocks	TCPIP MUCB TCPIP UCB TCPIP UDPSUM
TCPIP VMC	Display information about VMC and IUCV users	
TCPIP XCF	Display information about XCF links and dynamic VIPA	
ERRNO	Interpret error numbers	
ICMPHDR	Format an ICMP header	
IPHDR	Format an IP header	
RESOLVER	Format and summarize resolver control blocks	

Table 13. TCP/IP IPCS commands - showing the TCPIPICS commands first, followed by the general commands. (continued)

Command	Description	Alias
SETPRINT	Set destination so the IPCS subcommand output will be sent to a user ID or the printer	
SKMSG	Format a stream message	
TCPHDR	Format a TCP header	
TOD	Convert a S/390 64-bit time-of-day timestamp to a readable date and time	
UDPHDR	Format UDP header	

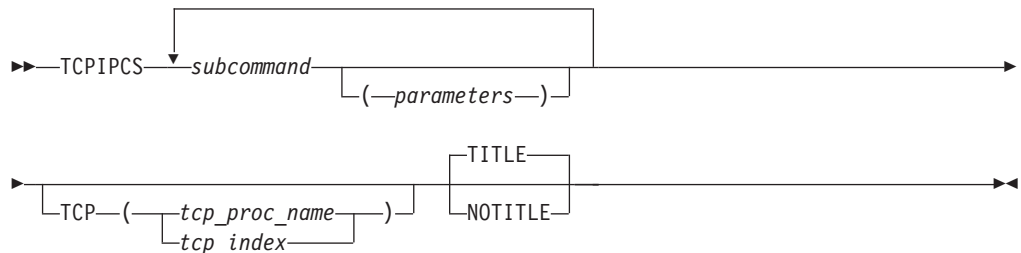
TCPIPICS

The following describes the TCPIPICS command.

Command syntax

The command syntax for all TCPIPICS subcommands includes an option to specify the TCP stack and to specify whether the title is displayed.

Note: For an explanation about syntax diagrams, see Appendix C, “How to read a syntax diagram” on page 729.



Parameters

The command parameters syntax for the TCPIPICS command is described below.

subcommand

Default is STATE.

parameters

Each subcommand has its own parameters.

- If a command has variable parameters, they can be omitted, specified as a single variable, or specified as a list. If no variable parameters are specified, an asterisk must be used as a placeholder if any keyword parameters are specified. If two or more variable parameters are specified, they must be enclosed in parentheses.
- To distinguish between the variable parameters, a parameter is assumed to be one of the following:
 - An index or small number if it is four digits or less, begins with zero to nine, and contains only hexadecimal digits (0–9, a–f, A–F). If a command

accepts multiple indices or small numbers, both are compared to the values and the first matching field is used.

- An address if it is more than four digits, begins with zero to nine, and contains only hexadecimal digits. For example, for the TCPIP CS DUAF command, both the DUCB and ASCB addresses of each DUCB are compared to the address parameter, and the first matching field is used to select the DUCB to display.
- An IPCS symbol name can also be specified for an address.
- Otherwise, the parameter is assumed to be a character string variable (such as TCP/IP procedure or job name, user ID, and command name).
- Keyword parameters can be in any order.
- If there are both keyword and variable parameters, all variable parameters must precede the keywords.

TCP

Specifies which TCP/IP stack is processed. The stack can be specified directly or indirectly. A stack can be specified directly by coding the **TCP** parameter with either *tcp_proc_name* or *tcp_index*. If no stack is specified directly, the output is reported for the stack with the lowest index matching the release of the TCPIP CS command. Once a particular stack is specified (whether specified directly or indirectly), that stack becomes the default. The stack index is saved as a symbol and is used as the default in future invocations of the TCPIP CS command. An alias for the **TCP** option is **PROC**.

Note:

All eight stack indices are available when TCP/IP starts, so any stack index can be selected. The existence of an index does not necessarily mean this stack can be formatted. If the stack was not included in the dump, then most of the information about a stack cannot be formatted. Most TCPIP control blocks are in the private TCP address space.

The fact that an index exists does not necessarily mean this stack index has ever been used. If you specify a stack index that has not been used, the version and release fields for this stack are zero, so you receive a message indicating the stack is not the same version and release as the TCPIP CS command:

Selected TCP/IP is not V1R4

tcp_proc_name

TCP/IP procedure name.

tcp_index

TCP/IP stack index (1–8).

TITLE

The title contains information about the dump and about the TCPIP CS command. By default, the title information is displayed. The title contains the following information.

- TCPIP CS command input parameters.
- Dump dataset name.
- Dump title.
- TSAB address.

- Table listing all TCP/IP stacks used in the dump and their TSEB address, stack index, procedure name, stack version, TSDB address, TSDX address, ASID, trace option bits, and stack status.
- Count of the number of TCP/IP stacks defined (used).
- Count of the number of active TCP/IP stacks found.
- Count of the number of active TCP/IP stacks matching the TCPIPICS command version and release.
- Procedure name and index of the stack being reported.

NOTITLE

Suppress the title lines. This is handy when you are processing lots of commands on the same dump and do not care to see the title information repeated.

Note: If you specify multiple keywords from the set {TITLE, NOTITLE}, only the last one is used.

Symbols defined

TCPIPICS defines the following IPCS symbols.

TSEBPTR

The address of the first TSEB control block.

TSEB_{*n*}

The address of the TSEB control block corresponding to the stack index *n*.

TCPIPICS subcommands

This section describes the available TCPIPICS Subcommands.

Note: Unless specifically shown otherwise, all examples in this chapter are valid for V1R4.

TCPIPICS API

Display information about the connects in the Sockets Extended Assembler Macro Application Programming Interface (Macro API) and the Pascal API.

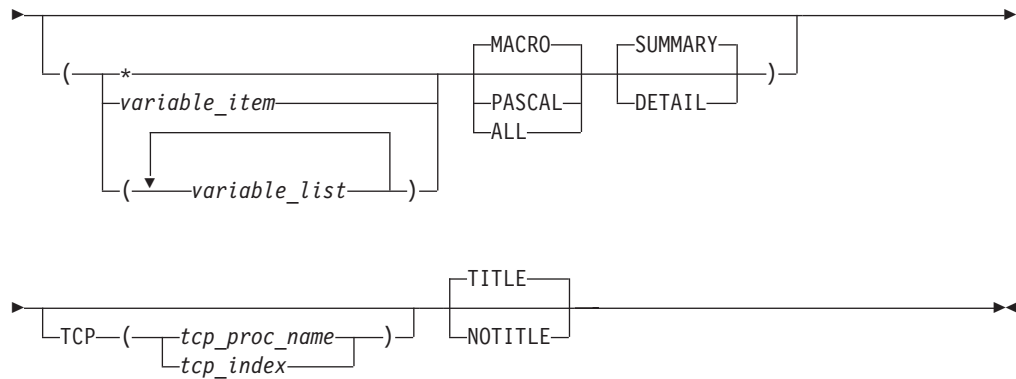
Note: The Macro API is the base for the CALL Instruction API, the CICS C API, and the CICS EZACICAL API. Refer to the *z/OS Communications Server: IP Application Programming Interface Guide* for more information about the native TCP/IP APIs.

Some API control blocks are in the application address space, which may not be available in the dump. If the application address space is available, the API control blocks will be formatted.

Syntax

Following is the syntax of the TCPIPICS API subcommand:

►►—TCPIPICS—API—————►



Parameters

If no parameters are specified, only information about the Macro API is summarized. Following are the parameters for the TCPIP API subcommand:

- * An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space, within parentheses:

Variable parameters are:

jobname

Displays only the API control blocks for this job name. The job name can be a TCP/IP application name or a stack name. Must contain from one to eight characters.

ASCB_address

Displays the API control blocks with this address space control block (ASCB) address. An IPCS symbol name can be specified for the address. The address is specified as one to eight hexadecimal digits. If an address begins with digit A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

ASID_number

Displays the API control blocks with this Address Space Identifier (ASID). The ASID is a hexadecimal number containing one to four digits.

In addition to the variable parameters described above, the following keyword parameters may be specified:

MACRO

Only display information for Macro APIs. MACRO is the default.

PASCAL

Only display information for Pascal APIs.

ALL

Display information for both APIs.

SUMMARY

Displays the addresses of the control blocks and other data in tables. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL also shows the contents of the control blocks.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {MACRO, PASCAL, ALL}, only the last one is used.
2. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

Sample output

The following is sample output of the TCPIPES API subcommand.

The contents of the SDST control blocks are formatted by the IPCS API subcommand if the DETAIL option is coded on the command (SUMMARY is the default and only the address of the SDST will be displayed in this case).

R14 Output:

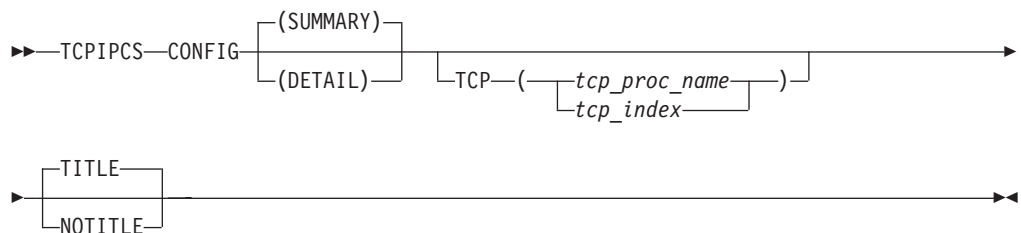
```
-- Array elements --
::
+00B2 SDST_LOCAL_IPADDRLEN. 00
+00B3 SDST_REMOTE_IPADDRLEN. 00
+00B4 SDST_LOCAL_IPADDR. 00000000 00000000 00000000 00000000
+00C4 SDST_REMOTE_IPADDR. 00000000 00000000 00000000 00000000
::
-- End of array --
```

TCPIPES CONFIG

Invocation of this subcommand displays each device interface, physical interface, and logical interface. The configuration summary table shows each logical interface with the name of its associated device and link.

Syntax

Following is the syntax of the TCPIPES CONFIG subcommand:



Parameters

Following are the parameters for the TCPIPES CONFIG subcommand:

SUMMARY

Displays each device, physical interface, and logical interface, and summarizes them all in one cross-reference table. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL also shows the interface cross-reference reports.

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIPCS CONFIG subcommand.

TCPIPCS CONFIG

Dataset: IPCS.R450697.V6TCBD1

Title: TCPCS2 CLIENT SIDE

The address of the TSAB is: 09DBE1A0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
09DBE1E0	1	TCPCS	V1R4	096C4000	096C40C8	0033	10841004	Active
09DBE260	2	TCPCS2	V1R4	096C9000	096C90C8	0034	10841004	Active

2 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

2 TCP/IP(s) for CS V1R4 found

=====

Analysis of Tcp/Ip for TCPCS2. Index: 2

Configuration control block summary

IPMAIN found at 095A83D0

IPMAIN6 found at 096CE470

Dif@	DeviceName	Next	Prev	DevR	DevW	Protocol
7F6AA408	LOOPBACK	7F1ED408	00000000	****	****	LOOPBACK
7F1ED408	OSAQDI03	00000000	7F6AA408	****	****	MPCIPA

IPv4	Pif@	LinkName	Next	Prev	DeviceName	Protocol	Dif@	Lif@
7F679468	LOOPBACK	7F503B88	00000000	LOOPBACK	LOOPBACK	7F6AA408	7F6792E8	
7F503B88	OSAQDI0L	00000000	7F679468	OSAQDI03	IPAQENET	7F1ED408	7F1ED008	

IPv6	Pif@	IntfName	Next	Prev	DeviceName	Protocol	Dif@	Lif@
7F503488	LOOPBACK6	7F503F08	00000000	LOOPBACK	LOOPBACK6	7F6AA408	7F3FDCE8	
7F503F08	OSAQDI26	00000000	7F503488	OSAQDI03	IPAQENET6	7F1ED408	7F6BF028	

IPv4	Lif@	LinkName	Next	Prev	Pif@	IpAddr
7F1ED008	OSAQDI0L	7F6792E8	00000000	7F503B88	9.67.115.82	
7F6792E8	LOOPBACK	00000000	7F1ED008	7F679468	127.0.0.1	

IPv6	Lif@	IntfName	Next	Prev	Pif@	IpAddr
7F3083C8	OSAQDI26	7F3FDCE8	00000000	7F503F08	FEC9:C2D4:1::9:67:115:82	
7F3FDCE8	LOOPBACK6	7F6BF028	7F3083C8	7F503488	::1	
7F6BF028	OSAQDI26	00000000	7F3FDCE8	7F503F08	FE80::2:559A:3F5F:1	

Configuration Summary

IPv4	Lif@	LinkName	DeviceName	DevR	DevW	IpAddr
7F1ED008	OSAQDI0L	OSAQDI03	****	****	9.67.115.82	
7F6792E8	LOOPBACK	LOOPBACK	****	****	127.0.0.1	

IPv6	Lif@	LinkName	DeviceName	DevR	DevW	IpAddr
7F3083C8	OSAQDI26	OSAQDI03	****	****	FEC9:C2D4:1::9:67:115:82	
7F3FDCE8	LOOPBACK6	LOOPBACK	****	****	::1	
7F6BF028	OSAQDI26	OSAQDI03	****	****	FE80::2:559A:3F5F:1	

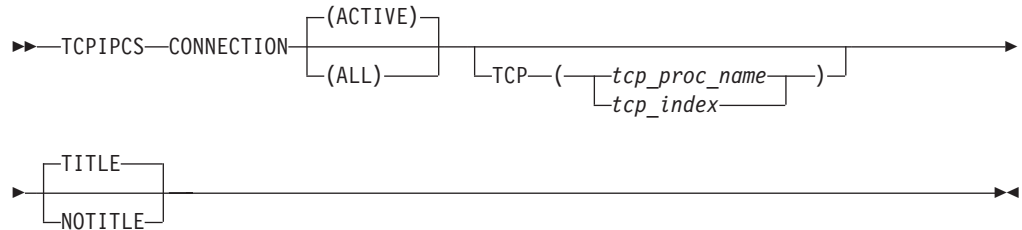
Analysis of Tcp/Ip for TCPCS2 completed

TCIPCS CONNECTION

Invocation of this command displays information about TCP, UDP, and raw connections. The information includes the user ID, connection ID, local IP address, foreign IP address, the connection state (for TCP connections only), and the protocol name (for raw connections only).

Syntax

Following is the syntax of the TCIPCS CONNECTION subcommand:



Parameters

Following are the parameters for the TCIPCS CONNECTION subcommand:

ACTIVE

Display only active connections. This is the default. Note that the number of connections reported for each protocol includes both inactive and active connections. Therefore the total may be higher than the number of displayed (active) connections.

ALL

Display all connections, regardless of state.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {ACTIVE, ALL}, only the last one is used.

Sample output

The following is a sample output of the TCIPCS CONNECTION subcommand.

Note: For the sample output shown, the default option is ACTIVE, so only active connections are shown. There are 10 total TCP connections, of which eight are active. There is one UDP connection, which is not active. Both raw connections are active.

```

TCIPCS CONNECTION
Dataset: IPCS.R8A0723.RASDUMP
Title: EZRPE005
The address of the TSAB is: 098221F0
Tseb SI Procedure Version TsdB TsdX Asid TraceOpts Status
09822230 1 TCPCS V1R4 08E85000 08E850C8 001E 9FFF7E7F Active
098222B0 2 TCPCS2 V1R4 08937000 089370C8 01F6 9FFF7E7F Active
2 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
2 TCP/IP(s) for CS V1R4 found
=====

```

Analysis of Tcp/Ip for TCPCS. Index: 1

TCP IPv4 Connections:

```

Userid Conn State Local Socket : 127.0.0.1..1024
TCPCS 00000012 Listening Foreign Socket: 0.0.0.0..0

```

```

BPX0INIT 00000019  Listening  Local Socket : 127.0.0.1..1024
                                Foreign Socket: 127.0.0.1..1025
TCPCS    00000016  Established Local Socket : 127.0.0.1..1024
                                Foreign Socket: 127.0.0.1..1025
TCPCS    00000014  Established Local Socket : 127.0.0.1..1024
                                Foreign Socket: 127.0.0.1..1025

4 TCP IPv4 connections

Active TCP IPv6 Connections:
Userid  Conn      State      Socket
FTPUNIX1 00000051 Listening   Local    ::0..21
                                Foreign    ::0..0
FTPMVS1  00000049 Listening   Local    ::0..1821
                                Foreign    ::0..0

2 TCP IPv6 connections

Active UDP Unicast IPv4 Connections:
Userid  Conn      Socket
PORTMAP 00000027 Local    0.0.0.0..111
                                Foreign  0.0.0.0..0
OSNMPD  00000030 Local    0.0.0.0..161
                                Foreign  127.0.0.1..162
MISCSRV 00000039 Local    198.11.98.124..7
                                Foreign  0.0.0.0..0
MISCSRV 0000003E Local    198.11.98.124..9
                                Foreign  0.0.0.0..0

4 UDP Unicast IPv4 connections

Active UDP Unicast IPv6 Connections:
Userid  Conn      Socket

0 UDP Unicast IPv6 connections

Active UDP Multicast IPv4 Connections:
Userid  Conn      Socket

0 UDP Multicast IPv4 connections

Active UDP Multicast IPv6 Connections:
Userid  Conn      Socket

0 UDP Multicast IPv6 connections

Active RAW Connections:
Userid  Conn      Protocol  Socket
TCPCS   00000006 IP          Local    0.0.0.0
                                Foreign  0.0.0.0
TCPCS   00000008 RAW         Local    0.0.0.0
                                Foreign  0.0.0.0
TCPCS   0000000E IP          Local    ::0
                                Foreign  ::0

3 RAW connections
Analysis of Tcp/Ip for TCPSVT completed

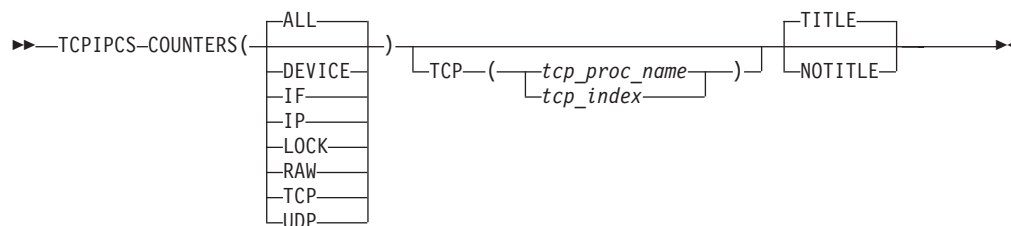
```

TCPIPCS COUNTERS

Invocation of this command displays information about TCP/IP internal execution statistics.

Syntax

Following is the syntax of the TCPIPCS COUNTERS subcommand:



Parameters

Following are the parameters for the TCIPCS COUNTERS subcommand:

ALL

Display all statistics.

DEVICE

Only display device statistics.

IF Only display IF layer statistics.

IP Only display IP layer statistics.

LOCK

Only display lock statistics.

RAW

Only display RAW layer statistics.

TCP

Only display TCP layer statistics.

UDP

Only display UDP layer statistics.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Sample output

The following is sample output of the TCIPCS COUNTERS subcommand for IP UDP.

```

| TCIPCS COUNTERS
| Dataset: IPCS.R8A0723.RASDUMP
| Title: EZRPE005
| The address of the TSAB is: 098221F0
| Tseb SI Procedure Version Tsd TsdX Asid TraceOpts Status
| 09822230 1 TCPCS V1R4 08E85000 08E850C8 001E 9FFF7E7F Active
| 098222B0 2 TCPCS2 V1R4 08937000 089370C8 01F6 9FFF7E7F Active
| 2 defined TCP/IP(s) were found
| 2 active TCP/IP(s) were found
| 2 TCP/IP(s) for CS V1R4 found
| =====
| Analysis of Tcp/Ip for TCPCS. Index: 1
| Statistics for Device LCS1 (DIF at 7F1F4408)
| LCSMsgs..... 0
| LCSVtCalls..... 0
| bac_active_array@... 7F1F4734
| bac_current_index... 0
| bgrp@..... 00000000
| bgrp_index..... 0
| saqa_next@..... 7F1F4834
| saqa_count..... 0
| max_tput..... 0
| sndbst_tput..... 0
| mbd..... 00000000 02710000
| oldest_msg..... 00000000 00000000

```

```

| sample_start..... B64C8AD5 06120C80
| sample_stop..... 00000000 00000000
| msgs..... 0
| clockwork..... 00000000 00000000
| opt_bf..... 0
| bkpt_bf..... 60
| prev_bf..... 0
| sndbst_bf..... 0
| next_experiment.... 00000000
| msg_sum..... 0
| bf..... 0
| blking_enabled..... 000003E7
| forced_flush..... 0
| ntxexp_mult..... 1
| blkalgo_state..... 00
| tput_samples..... 0
| avg_tput..... 0
| bf_misc2..... 00
| optbf_shaky..... 0
| bkpt_tput..... 0
| prev_tput..... 0
| shortsamp_start.... B64C90ED C3E70681
| blk_diag_seq..... 16,232
| blk_diag_next..... 0
| blk_diag_bfe(1).... 0
| blk_diag_bfe(2).... 0
| blk_diag_bfe(3).... 0
| blk_diag_bfe(4).... 0
| blk_diag_bfe(5).... 0
| blk_diag_bfe(6).... 0
| blk_diag_bfe(7).... 0
| blk_diag_bfe(8).... 0
| blk_diag_bfe(9).... 0
| blk_diag_bfe(10).... 0
| blk_diag_bfe(11).... 0
| blk_diag_bfe(12).... 0
| blk_diag_bfe(13).... 0
| blk_diag_bfe(14).... 0
| blk_diag_bfe(15).... 0
| blk_diag_sqe(1).... 0
| blk_diag_sqe(2).... 0
| blk_diag_sqe(3).... 0
| blk_diag_sqe(4).... 0
| blk_diag_sqe(5).... 0
| blk_diag_sqe(6).... 0
| blk_diag_sqe(7).... 0
| blk_diag_sqe(8).... 0
| blk_diag_sqe(9).... 0
| blk_diag_sqe(10).... 0
| blk_diag_sqe(11).... 0
| blk_diag_sqe(12).... 0
| blk_diag_sqe(13).... 0
| blk_diag_sqe(14).... 0
| blk_diag_sqe(15).... 0
| tputarr(0)..... 0
| tputarr(1)..... 0
| tputarr(2)..... 0
| tputarr(3)..... 0
| tputarr(4)..... 0
| tputarr(5)..... 0
| tputarr(6)..... 0
| tputarr(7)..... 0
| tputarr(8)..... 0
| tputarr(9)..... 0
| tputarr(10)..... 0
| tputarr(11)..... 0
| tputarr(12)..... 0

```

```

tputarr(13)..... 0
tputarr(14)..... 0
tputarr(15)..... 0
tputarr(16)..... 0
tputarr(17)..... 0
tputarr(18)..... 0
tputarr(19)..... 0
tputarr(20)..... 0
tputarr(21)..... 0
tputarr(22)..... 0
tputarr(23)..... 0
tputarr(24)..... 0
tputarr(25)..... 0
tputarr(26)..... 0
tputarr(27)..... 0
tputarr(28)..... 0
tputarr(29)..... 0
tputarr(30)..... 0
tputarr(31)..... 0
nbflush1..... 0
nbflush2..... 0
tmrcall..... 0
failedexp..... 0
bfeperf..... 0
magic_bndry..... 0
callbp1..... 0
pdusbp1..... 0
rfifbp1..... 0
rfi2bp1..... 0
raspbp1..... 0
mwrcbp1..... 0
xbgrbp1..... 0
callout..... 0
bglkd..... 0
bgulkd..... 0
callvtm..... 0
Statistics for Device TRLE1A (DIF at 7F1F1008)
bac_active_array@... 7F1F1334
bac_current_index... 0
bgrp@..... 00000000
bgrp_index..... 0
saqa_next@..... 7F1F1434
saqa_count..... 0
max_tput..... 0
sndbst_tput..... 0
mbd..... 00000000 02710000
oldest_msg..... 00000000 00000000
sample_start..... B64C8AFA E964670A
sample_stop..... 00000000 00000000
msgs..... 0
clockwork..... 00000000 00000000
opt_bf..... 0
bkpt_bf..... 60
prev_bf..... 0
sndbst_bf..... 0
next_experiment.... 00000000
msg_sum..... 0
bf..... 0
blking_enabled..... 000003E7
forced_flush..... 0
nxtexp_mult..... 1
blkalgo_state..... 00
tput_samples..... 0
avg_tput..... 0
bf_misc2..... 40
optbf_shaky..... 0
bkpt_tput..... 0

```

```

|      prev_tput..... 0
|      shortsamp_start.... B64C90ED C3E728C1
|      blk_diag_seq..... 15,841
|      blk_diag_next..... 0
|      blk_diag_bfe(1).... 0
|      blk_diag_bfe(2).... 0
|      blk_diag_bfe(3).... 0
|      blk_diag_bfe(4).... 0
|      blk_diag_bfe(5).... 0
|      blk_diag_bfe(6).... 0
|      blk_diag_bfe(7).... 0
|      blk_diag_bfe(8).... 0
|      blk_diag_bfe(9).... 0
|      blk_diag_bfe(10).... 0
|      blk_diag_bfe(11).... 0
|      blk_diag_bfe(12).... 0
|      blk_diag_bfe(13).... 0
|      blk_diag_bfe(14).... 0
|      blk_diag_bfe(15).... 0
|      blk_diag_sqe(1).... 0
|      blk_diag_sqe(2).... 0
|      blk_diag_sqe(3).... 0
|      blk_diag_sqe(4).... 0
|      blk_diag_sqe(5).... 0
|      blk_diag_sqe(6).... 0
|      blk_diag_sqe(7).... 0
|      blk_diag_sqe(8).... 0
|      blk_diag_sqe(9).... 0
|      blk_diag_sqe(10).... 0
|      blk_diag_sqe(11).... 0
|      blk_diag_sqe(12).... 0
|      blk_diag_sqe(13).... 0
|      blk_diag_sqe(14).... 0
|      blk_diag_sqe(15).... 0
|      tputarr(0)..... 0
|      tputarr(1)..... 0
|      tputarr(2)..... 0
|      tputarr(3)..... 0
|      tputarr(4)..... 0
|      tputarr(5)..... 0
|      tputarr(6)..... 0
|      tputarr(7)..... 0
|      tputarr(8)..... 0
|      tputarr(9)..... 0
|      tputarr(10)..... 0
|      tputarr(11)..... 0
|      tputarr(12)..... 0
|      tputarr(13)..... 0
|      tputarr(14)..... 0
|      tputarr(15)..... 0
|      tputarr(16)..... 0
|      tputarr(17)..... 0
|      tputarr(18)..... 0
|      tputarr(19)..... 0
|      tputarr(20)..... 0
|      tputarr(21)..... 0
|      tputarr(22)..... 0
|      tputarr(23)..... 0
|      tputarr(24)..... 0
|      tputarr(25)..... 0
|      tputarr(26)..... 0
|      tputarr(27)..... 0
|      tputarr(28)..... 0
|      tputarr(29)..... 0
|      tputarr(30)..... 0
|      tputarr(31)..... 0
|      nbflush1..... 0

```



```

|      nbflush2..... 0
|      tmrcall..... 0
|      failedexp..... 0
|      bfeperf..... 0
|      magic_bndry..... 0
|      callbp1..... 0
|      pdusbp1..... 0
|      rfifbp1..... 0
|      rfi2bp1..... 0
|      raspbp1..... 0
|      mwrcbp1..... 0
|      xbgrbp1..... 0
|      callout..... 0
|      bglkd..... 0
|      bgulkd..... 0
|      callvtm..... 0
|
|      Statistics Extension for Device TRLE1A (DIF at 7F1F1008)
|      bac_active_array@... 00000000
|      bac_current_index... 0
|      bgrp@..... 00000000
|      bgrp_index..... 0
|      saqa_next@..... 00000000
|      saqa_count..... 0
|      max_tput..... 0
|      sndbst_tput..... 0
|      mbd..... 00000000 02710000
|      oldest_msg..... 00000000 00000000
|      sample_start..... B64C8AFA E963CD8A
|      sample_stop..... 00000000 00000000
|      msgs..... 0
|      clockwork..... 00000000 00000000
|      opt_bf..... 0
|      bkpt_bf..... 60
|      prev_bf..... 0
|      sndbst_bf..... 0
|      next_experiment..... 00000000
|      msg_sum..... 0
|      bf..... 0
|      blking_enabled..... 000003E7
|      forced_flush..... 0
|      ntxexp_mult..... 1
|      blkalgo_state..... 00
|      tput_samples..... 0
|      avg_tput..... 0
|      bf_misc2..... 00
|      optbf_shaky..... 0
|      bkpt_tput..... 0
|      prev_tput..... 0
|      shortsamp_start..... B64C90ED C3E66941
|      blk_diag_seq..... 15,841
|      blk_diag_next..... 0
|      blk_diag_bfe(1)..... 0
|      blk_diag_bfe(2)..... 0
|      blk_diag_bfe(3)..... 0
|      blk_diag_bfe(4)..... 0
|      blk_diag_bfe(5)..... 0
|      blk_diag_bfe(6)..... 0
|      blk_diag_bfe(7)..... 0
|      blk_diag_bfe(8)..... 0
|      blk_diag_bfe(9)..... 0
|      blk_diag_bfe(10)..... 0
|      blk_diag_bfe(11)..... 0
|      blk_diag_bfe(12)..... 0
|      blk_diag_bfe(13)..... 0
|      blk_diag_bfe(14)..... 0
|      blk_diag_bfe(15)..... 0

```

```

blk_diag_sqe(1)..... 0
blk_diag_sqe(2)..... 0
blk_diag_sqe(3)..... 0
blk_diag_sqe(4)..... 0
blk_diag_sqe(5)..... 0
blk_diag_sqe(6)..... 0
blk_diag_sqe(7)..... 0
blk_diag_sqe(8)..... 0
blk_diag_sqe(9)..... 0
blk_diag_sqe(10).... 0
blk_diag_sqe(11).... 0
blk_diag_sqe(12).... 0
blk_diag_sqe(13).... 0
blk_diag_sqe(14).... 0
blk_diag_sqe(15).... 0
tputarr(0)..... 0
tputarr(1)..... 0
tputarr(2)..... 0
tputarr(3)..... 0
tputarr(4)..... 0
tputarr(5)..... 0
tputarr(6)..... 0
tputarr(7)..... 0
tputarr(8)..... 0
tputarr(9)..... 0
tputarr(10)..... 0
tputarr(11)..... 0
tputarr(12)..... 0
tputarr(13)..... 0
tputarr(14)..... 0
tputarr(15)..... 0
tputarr(16)..... 0
tputarr(17)..... 0
tputarr(18)..... 0
tputarr(19)..... 0
tputarr(20)..... 0
tputarr(21)..... 0
tputarr(22)..... 0
tputarr(23)..... 0
tputarr(24)..... 0
tputarr(25)..... 0
tputarr(26)..... 0
tputarr(27)..... 0
tputarr(28)..... 0
tputarr(29)..... 0
tputarr(30)..... 0
tputarr(31)..... 0
nbflush1..... 0
nbflush2..... 0
tmrcall..... 0
failedexp..... 0
bfeperf..... 0
magic_bndry..... 0
callbp1..... 0
pdusbpl..... 0
rfifbp1..... 0
rfi2bp1..... 0
raspbpl..... 0
mwrcbp1..... 0
xbgrbp1..... 0
callout..... 0
bglkd..... 0
bgulkd..... 0
callvtm..... 0
IF Statistics
hgram(1)..... 0
hgram(2)..... 0

```

hgram(3).....	0
hgram(4).....	0
hgram(5).....	0
hgram(6).....	0
hgram(7).....	0
hgram(8).....	0
hgram(9).....	0
hgram(10).....	0
hgram(11).....	0
hgram(12).....	0
hgram(13).....	0
hgram(14).....	0
hgram(15).....	0
hgram(16).....	0
nummsg.....	0
bkrace1.....	0
bkrace2.....	0
extfail.....	0
oberror.....	0
incsm.....	0
tmrcall.....	0
iutmuc.....	0
maxbf.....	61
extxbf.....	0
unperr.....	0
unpslim.....	2,048
unpbndy.....	0
pksz.....	2,048
badmdata.....	0
mpcin2.....	0
lcscm1.....	0
lcscm2.....	0
unpsimp.....	0
unpcplx.....	0
fcsms1.....	33
fcsms2.....	0
bkrace3.....	0
unp24.....	0
bkreenab.....	0
pckcahit.....	0
pckcamis.....	0
iutsend.....	17
frcafix.....	0
baddest.....	0
unpsmpct.....	60
unplgpct.....	10
unpminpkts.....	15
unpsmsz.....	512
unplgsz.....	2,048
bgrpfail.....	0
bgrpcntr.....	0
inbcalls.....	764
IP Statistics	
nonbat.....	242
batch.....	0
batnum.....	0
nonrsm.....	0
batrsm.....	0
rsmnum.....	0
rteadd.....	24
rtedel.....	0
rteinc.....	33
rtedec.....	31
rtpadd.....	8
rtpdel.....	0
rtechg.....	56
trredr.....	0

```

|      trsusp..... 0
|      trsust..... 0
|      dupfrg..... 0
|      dataadj1..... 0
|      dataadj2..... 0
| IP6 Statistics
|      nonbat..... 10
|      batch..... 0
|      batnum..... 0
|      nonrsm..... 0
|      batrsm..... 0
|      rsmnum..... 0
|      rteadd..... 0
|      rtedel..... 0
|      rteinc..... 7
|      rtedec..... 7
|      rtpadd..... 16
|      rtpdel..... 0
|      rtechg..... 109
|      trredr..... 0
|      trsusp..... 0
|      trsust..... 0
|      dupfrg..... 0
|      dataadj1..... 0
|      dataadj2..... 0
|      lifdel..... 0
|      noreclaim..... 0
|      hdrpullup..... 0
|      dadfailtot..... 0
|      dadfailll..... 0
| Lock Statistics
| Class 1 Level 1 Name STREAMACCESS
|      obtain_excl..... 0
|      obtain_shr..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_shr..... 0
|      delay..... 0
| Class 1 Level 2 Name STREAMSLEEP
|      obtain_excl..... 0
|      obtain_shr..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_shr..... 0
|      delay..... 0
| Class 1 Level 3 Name STREAMROOT
|      obtain_excl..... 0
|      obtain_shr..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_shr..... 0
|      delay..... 0
| Class 2 Level 1 Name MultipleUse
|      obtain_excl..... 0
|      obtain_shr..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_shr..... 0
|      delay..... 0
| Class 3 Level 1 Name IPMAIN
|      obtain_excl..... 0
|      obtain_shr..... 0
|      cont_srb..... 27

```

```

|         cont_tcb..... 2
|         Dup_excl..... 0
|         Dup_shr..... 6
|         delay..... 0
| Class 3 Level 2 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 3 Level 3 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 3 Level 4 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 3 Level 5 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 3 Level 6 Name IDM_POLICY
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 3 Level 7 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 3 Level 8 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 3 Level 9 Name PCT
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0

```

```

|      delay..... 0
|      Class 4 Level 1 Name UCB
|      obtain_excl..... 0
|      obtain_srb..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_srb..... 0
|      delay..... 0
|      Class 4 Level 2 Name MUCB
|      obtain_excl..... 0
|      obtain_srb..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_srb..... 0
|      delay..... 0
|      Class 5 Level 1 Name MRCB
|      obtain_excl..... 0
|      obtain_srb..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_srb..... 0
|      delay..... 0
|      Class 5 Level 2 Name RCB
|      obtain_excl..... 0
|      obtain_srb..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_srb..... 0
|      delay..... 0
|      Class 6 Level 1 Name MTCB
|      obtain_excl..... 0
|      obtain_srb..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_srb..... 0
|      delay..... 0
|      Class 6 Level 2 Name TCB
|      obtain_excl..... 0
|      obtain_srb..... 0
|      cont_srb..... 2
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_srb..... 0
|      delay..... 0
|      Class 6 Level 3 Name TCB2
|      obtain_excl..... 0
|      obtain_srb..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_srb..... 0
|      delay..... 0
|      Class 6 Level 4 Name MultipleUse
|      obtain_excl..... 0
|      obtain_srb..... 0
|      cont_srb..... 0
|      cont_tcb..... 0
|      Dup_excl..... 0
|      Dup_srb..... 0
|      delay..... 0
|      Class 7 Level 1 Name FWEONE
|      obtain_excl..... 0

```

```

|         obtain_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_srb..... 0
|         delay..... 0
| Class 7 Level 2 Name FWETWO
|         obtain_excl..... 0
|         obtain_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_srb..... 0
|         delay..... 0
| Class 7 Level 3 Name FWETHREE
|         obtain_excl..... 0
|         obtain_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_srb..... 0
|         delay..... 0
| Class 8 Level 1 Name MultipleUse
|         obtain_excl..... 0
|         obtain_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_srb..... 0
|         delay..... 0
| Class 9 Level 1 Name XCF
|         obtain_excl..... 0
|         obtain_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_srb..... 0
|         delay..... 0
| Class 9 Level 2 Name MultipleUse
|         obtain_excl..... 0
|         obtain_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_srb..... 0
|         delay..... 0
| Class 9 Level 3 Name XFWLM
|         obtain_excl..... 0
|         obtain_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_srb..... 0
|         delay..... 0
| Class 10 Level 1 Name MultipleUse
|         obtain_excl..... 0
|         obtain_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_srb..... 0
|         delay..... 0
| Class 11 Level 1 Name SASPQLOCK
|         obtain_excl..... 0
|         obtain_srb..... 0
|         cont_tcb..... 0

```

```

|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 11 Level 2 Name NOTICEQLOCK
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 12 Level 1 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 12 Level 2 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 12 Level 3 Name TCVB
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 12 Level 4 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 12 Level 5 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 12 Level 6 Name MultipleUse
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0
| Class 12 Level 7 Name TCFG_CLEANUP
|         obtain_excl..... 0
|         obtain_shr..... 0
|         cont_srb..... 0
|         cont_tcb..... 0
|         Dup_excl..... 0
|         Dup_shr..... 0
|         delay..... 0

```



```

Class 12 Level 8 Name LU_MAP
  obtain_excl..... 0
  obtain_shr..... 0
  cont_srb..... 0
  cont_tcb..... 0
  Dup_excl..... 0
  Dup_shr..... 0
  delay..... 0
Class 12 Level 9 Name LU_APP
  obtain_excl..... 0
  obtain_shr..... 0
  cont_srb..... 0
  cont_tcb..... 0
  Dup_excl..... 0
  Dup_shr..... 0
  delay..... 0
RAW Statistics
None Available.
TCP Statistics
  tcp_ActiveOpens..... 3
  tcp_PassiveOpens..... 8
  tcp_AttemptFails..... 0
  tcp_EstabResets..... 0
  tcp_CurrEstab..... 2
  tcp_InSegs..... 42
  tcp_OutSegs..... 45
  tcp_RetransSegs..... 0
  tcp_InErrs..... 0
  tcp_OutRsts..... 2
  tcp_PassiveDrops.... 5
  tcp_TimeWaitReused.. 0
  tcp_ConnClosed..... 20
  tcp_PredictAck..... 0
  tcp_PredictData..... 9
  tcp_InDupAck..... 9
  tcp_InBadSum..... 0
  tcp_InBadLen..... 0
  tcp_InShort..... 0
  tcp_InPawsDrop..... 0
  tcp_InAllBeforeWin.. 0
  tcp_InSomeBeforeWin. 0
  tcp_InAllAfterWin... 0
  tcp_InSomeAfterWin.. 0
  tcp_InOutOfOrder.... 0
  tcp_InAfterClose.... 0
  tcp_InWinProbes..... 0
  tcp_InWinUpdates.... 0
  tcp_OutWinUpdates... 0
  tcp_OutDelayAcks.... 9
  tcp_RxmtTimers..... 0
  tcp_RxmtDrops..... 0
  tcp_PMTURxmts..... 0
  tcp_PMTUErrors..... 0
  tcp_OutWinProbes.... 0
  tcp_ProbeDrops..... 0
  tcp_KeepAliveProbes. 0
  tcp_KeepAliveDrops.. 0
  tcp_Finwait2Drops... 0
  tcp_dropit(0)..... 0
  tcp_dropit(1)..... 0
  tcp_dropit(2)..... 0
  tcp_dropit(3)..... 0
  tcp_dropit(4)..... 0
  tcp_dropit(5)..... 0
  tcp_dropit(6)..... 0
  tcp_dropit(7)..... 2
  tcp_dropit(8)..... 0

```

```

|          tcp_dropit(9)..... 0
|          tcp_dropit(10)..... 0
|          tcp_dropit(11)..... 0
|          tcp_dropit(12)..... 0
|          tcp_dropit(13)..... 0
|          tcp_dropit(14)..... 0
|          tcp_dropit(15)..... 0
|          tcp_dropit(16)..... 0
|          tcp_dropit(17)..... 0
|          tcp_dropit(18)..... 0
|          tcp_dropit(19)..... 0
|          tcp_dropit(20)..... 0
|          tcp_dropit(21)..... 0
|          tcp_dropit(22)..... 0
|          tcp_dropit(23)..... 0
|          tcp_batches..... 42
|          tcp_contention_avoid 0
|          tcp_tcb_find_fail... 3
|          tcp_tcb_alloc_fail.. 0
|          tcp_oos_alloc..... 0
|          tcp_oos_fast..... 0
|          tcp_oos_slow..... 0
|          tcp_accept_wait..... 0
|          tcp_write_wait..... 0
|          tcp_write_error..... 0
|          tcp_probe_timer..... 0
|          tcp_delayack_timer.. 9
|          tcp_keepalive_timer. 0
|          tcp_linger_timer.... 0
|          tcp_timewait_timer.. 7
|          tcp_finwait2_timer.. 0
|          itS_tcp_FRCA..... 00000000
|          itS_tcp_Floge..... 00000000
|          itS_tcp_frca_fixedli 0
| UDP Statistics
|          rd..... 83
|          rdnum..... 83
|          batch..... 0
|          nonbat..... 83
| Analysis of Tcp/Ip for TCPCS completed

```

TCPIPCS DUAF

Invocation of this command displays a summary of each dispatchable unit control block (DUCB). Each entry in the dispatchable unit allocation table (DUAT) points to a DUCB. The DUAT entry contains the status of the DUCB and identifies the ASID with which the DUCB is associated. If no parameters are specified, the output contains a summary of the DUAT, followed by a summary of each DUCB.

The status of each DUCB is abbreviated:

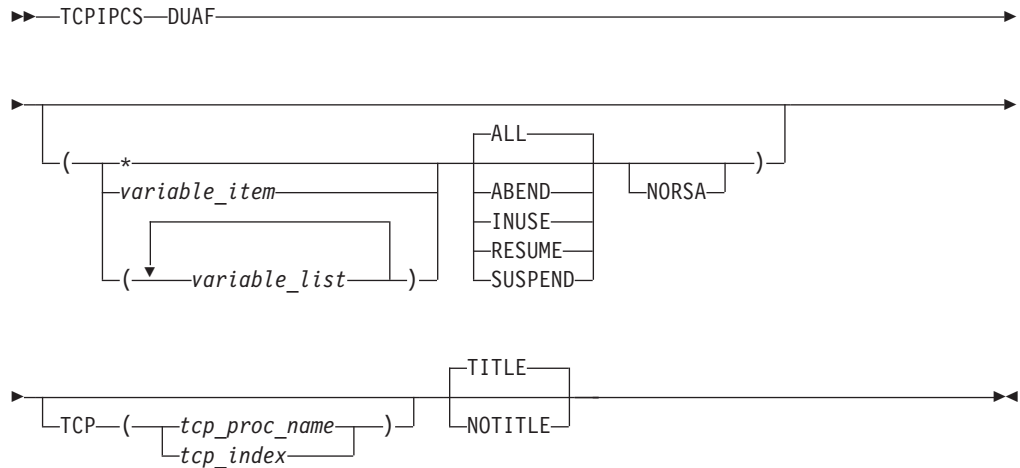
Ab The DUCB has ABENDED.
lu The DUCB is in use.
Re The DUCB is in resume state.
Su The DUCB has been suspended.

The DUCB status may be followed by the recovery stack. There is one line for each register save area (RSA) found in the DUCB (and its DUSA extension, if present). The address of each RSA, its previous pointer, its next pointer, and the module name are shown.

A register save area displayed as RSA* indicates that the RSA is not in the active chain. If all RSAs are shown like this, the DUCB is not in use.

Syntax

Following is the syntax of the TCPIPICS DUAF subcommand:



Parameters

If no parameters are specified, all active DUCBs are summarized. Following are the parameters for the TCPIPICS DUAF command:

- * An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space, within parentheses:

jobname

Displays only the DUCBs with this job name. The job name can be a TCP/IP application name or a stack name. Must contain from one to eight characters.

DUCB_address

Displays the DUCB with this address. An IPCS symbol name can be specified for the address. The address is specified as one to eight hexadecimal digits. If an address begins with characters A–F, prefix the address with a 0 to avoid the address being interpreted as a symbol name or as a character string.

DUCB_index

Displays this DUCB with this index. The index is a hexadecimal number containing one to four digits. The lowest index is 0. If an address begins with characters A–F, prefix the address with a 0 to avoid the address being interpreted as a symbol name or as a character string.

ASCB_address

Displays the DUCB with this address space control block (ASCB) address. An IPCS symbol name can be specified for the address. The address is specified as one to eight hexadecimal digits. If an address begins with characters A–F, prefix the address with a 0 to avoid the address being interpreted as a symbol name or as a character string.

ASID_number

Displays the DUCB with this ASID. The ASID is a hexadecimal number containing one to four digits.

In addition to the variable parameters described above, the following keyword parameters can be specified:

ALL

Display information for all active DUCBs. This is the default.

ABEND

Only display information for DUCBs that ABENDED.

INUSE

Only display information for DUCBs currently being used

RESUME

Only display information for DUCBs that are resumed.

SUSPEND

Only display information for DUCBs that are suspended.

NORSA

Do not display the contents of the DUCBs' register save areas (RSA). By default, the RSA contents are displayed.

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {ALL, ABEND, INUSE, RESUME, SUSPEND}, only the last one will be used.

Sample output

The following is a sample output of the TCPIPICS DUAF subcommand.

```
TCPIPICS DUAF( (0876C000 0B) INUSE )
Dataset: IPCS.A594094.DUMPK
Title:  TCPCS   V2R10: Job(USER15 ) EZBITRAC(HTCP50A 99.266)+
       000304 S0C4/00000004 TCB P=0029,S=000E,H=0019
```

The address of the TSAB is: 08D138C0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
08D13900	1	TCPCS	V2R10	0885A000	0885A0C8	0029	9FFFFFF7F	Active

1 defined TCP/IP(s) were found

1 active TCP/IP(s) were found

1 TCP/IP(s) for CS V2R10 found

=====

Analysis of Tcp/Ip for TCPCS. Index: 1

Dispatchable Unit Summary

INDEX	DUAE	DUCB	DUSA	ASCB	ASID	JOBNAME	ABEND	STATUS
10000003	08859040	0876C000	0876C100	00FB7080	0029	TCPCS	00000000	Iu
RSA 0876C3F8 Prev 00005D98 Next 0876C8C0 Mod EZBIE0ER								
RSA* 0876C8C8 Prev 0876C3F8 Next 00000000 Mod EZBITSTO								
1384 bytes were used								
1000000B	08859080	08784000	08784100	00FB7980	0019	USER15	000C4000	Ab Iu

```

RSA 087843F8 Prev 09BB9798 Next 087846B8 Mod EZBPFSOC
RSA 087846C0 Prev 087843F8 Next 08784988 Mod EZBPFOPN
RSA 08784990 Prev 087846C0 Next 08784DB0 Mod EZBUDSTR
RSA 08784DB8 Prev 08784990 Next 087855A8 Mod EZBITRAC
4536 bytes were used

```

```

82 DU control blocks were found
12 DU control blocks were in use
0 DU control blocks were suspended
0 DU control blocks were resumed
1 DU control blocks had abended
2 DU control blocks were formatted

```

The maximum DUCB size found is 4536 bytes

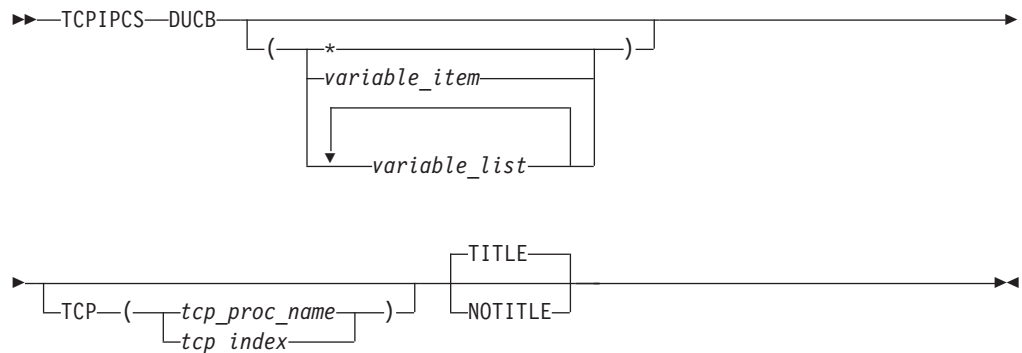
Analysis of Tcp/Ip for TCPCS completed

TCPIPCS DUCB

Invocation of this command displays the contents of each dispatchable unit control block (DUCB). Each entry in the dispatchable unit allocation table (DUAT) points to a DUCB. The DUAT entry contains the status of the DUCB and identifies the ASID with which the DUCB is associated. In the output, the DUAT is summarized. The contents of each DUCB are then displayed, followed by each DUSA for the DUCB. The first dispatchable unit stack area (DUSA) is followed by information from each register save area (RSA). Each register from the RSA is listed, showing its address and offset from the other registers in the register save area. The address of the parameter list (pointed to by R1) and the first five words at that address are also given. Each RSA is formatted. The recovery stack is also displayed.

Syntax

Following is the syntax of the TCPIPCS DUCB subcommand:



Parameters

Following are the parameters for the TCPIPCS DUCB subcommand:

If no parameters are specified, all DUCBs are displayed.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space:

jobname

Displays only the DUCBs with this job name. The job name can be a TCP/IP application name or a stack name. Must contain from one to eight characters.

DUCB_address

Displays the DUCB with this address. An IPCS symbol name can be specified for the address. The address is specified as one to eight hexadecimal digits. If an address begins with digit A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

DUCB_index

Displays this DUCB with this index. The index is a hexadecimal number containing one to four digits. The lowest index is zero.

ASCB_address

Displays the DUCB with this address space control block address (ASCB). An IPCS symbol name may be specified for the address. The address is specified as one to eight hexadecimal digits. If an address begins with digit A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

asid_number

Displays the DUCB with this ASID. The ASID is a hexadecimal number containing one to four digits.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Sample output

The following is sample output of the TCPIPES DUCB subcommand:

```
TCPIPES DUCB
Dataset: IPCS.R8A0723.RASDUMP
Title: EZRPE005
The address of the TSAB is: 098221F0
Tseb SI Procedure Version TsdX Asid TraceOpts Status
09822230 1 TCPCS V1R4 08E85000 08E850C8 001E 9FFF7E7F Active
098222B0 2 TCPCS2 V1R4 08937000 089370C8 01F6 9FFF7E7F Active
  2 defined TCP/IP(s) were found
  2 active TCP/IP(s) were found
  2 TCP/IP(s) for CS V1R4 found
=====
Analysis of Tcp/Ip for TCPCS. Index: 1
DUCB Detail Analysis
Dispatchable Unit Allocation Table: 08E83010
+0000 DUAT0 EYE..... DUAT NEXT..... 00000000
+0018 DUAE0 DUCB..... 08D8D010 FLAGS.... 0014001E
+0020 DUAE1 DUCB..... 08D90000 FLAGS.... 0034001E
+0028 DUAE2 DUCB..... 08D93000 FLAGS.... 0050001E
+0030 DUAE3 DUCB..... 08D96000 FLAGS.... 0014001E
+0038 DUAE4 DUCB..... 08D99000 FLAGS.... 5490001E
+0040 DUAE5 DUCB..... 08D9C000 FLAGS.... 01C0001E
+0048 DUAE6 DUCB..... 08D9F000 FLAGS.... 0070001E
+0050 DUAE7 DUCB..... 08DA2000 FLAGS.... 00C4001E
+0058 DUAE8 DUCB..... 08DA5000 FLAGS.... 0194001E
+0060 DUAE9 DUCB..... 08DA8000 FLAGS.... 08D4001E
+0068 DUAE10 DUCB..... 08DAB000 FLAGS.... 0284001E
+0070 DUAE11 DUCB..... 08DAE000 FLAGS.... 00B4001E
+0078 DUAE12 DUCB..... 08DB1000 FLAGS.... 00000000
+0080 DUAE13 DUCB..... 08DB4000 FLAGS.... 00000000
+0088 DUAE14 DUCB..... 08DB7000 FLAGS.... 00000000
+0090 DUAE15 DUCB..... 08DBA000 FLAGS.... 00000000
+0098 DUAE16 DUCB..... 08DBD000 FLAGS.... 00000000
+00A0 DUAE17 DUCB..... 08DC0000 FLAGS.... 00000000
+00A8 DUAE18 DUCB..... 08DC3000 FLAGS.... 00000000
+00B0 DUAE19 DUCB..... 08DC6000 FLAGS.... 00000000
+00B8 DUAE20 DUCB..... 08DC9000 FLAGS.... 00000000
+00C0 DUAE21 DUCB..... 08DCC000 FLAGS.... 00000000
+00C8 DUAE22 DUCB..... 08DCF000 FLAGS.... 00000000
+00D0 DUAE23 DUCB..... 08DD2000 FLAGS.... 00000000
```

```

+00D8 DUA24      DUCB..... 08DD5000  FLAGS.... 00000000
+00E0 DUA25      DUCB..... 08DD8000  FLAGS.... 00000000
+00E8 DUA26      DUCB..... 08DDB000  FLAGS.... 00000000
+00F0 DUA27      DUCB..... 08DDE000  FLAGS.... 00000000
+00F8 DUA28      DUCB..... 08DE1000  FLAGS.... 00000000
+0100 DUA29      DUCB..... 08DE4000  FLAGS.... 00000000
+0108 DUA30      DUCB..... 08DE7000  FLAGS.... 00000000
+0110 DUA31      DUCB..... 08DEA000  FLAGS.... 00000000
+0118 DUA32      DUCB..... 08DED000  FLAGS.... 00000000
+0120 DUA33      DUCB..... 08DF0000  FLAGS.... 00000000
+0128 DUA34      DUCB..... 08DF3000  FLAGS.... 00000000
+0130 DUA35      DUCB..... 08DF6000  FLAGS.... 00000000
+0138 DUA36      DUCB..... 08DF9000  FLAGS.... 00000000
+0140 DUA37      DUCB..... 08DFC000  FLAGS.... 00000000
+0148 DUA38      DUCB..... 08DFF000  FLAGS.... 00000000
+0150 DUA39      DUCB..... 08E02000  FLAGS.... 00000000
+0158 DUA40      DUCB..... 08E05000  FLAGS.... 00000000
+0160 DUA41      DUCB..... 08E08000  FLAGS.... 00000000
+0168 DUA42      DUCB..... 08E0B000  FLAGS.... 00000000
+0170 DUA43      DUCB..... 08E0E000  FLAGS.... 00000000
+0178 DUA44      DUCB..... 08E11000  FLAGS.... 00000000
+0180 DUA45      DUCB..... 08E14000  FLAGS.... 00000000
+0188 DUA46      DUCB..... 08E17000  FLAGS.... 00000000
+0190 DUA47      DUCB..... 08E1A000  FLAGS.... 00000000
+0198 DUA48      DUCB..... 08E1D000  FLAGS.... 00000000
+01A0 DUA49      DUCB..... 08E20000  FLAGS.... 00000000
+01A8 DUA50      DUCB..... 08E23000  FLAGS.... 00000000
+01B0 DUA51      DUCB..... 08E26000  FLAGS.... 00000000
+01B8 DUA52      DUCB..... 08E29000  FLAGS.... 00000000
+01C0 DUA53      DUCB..... 08E2C000  FLAGS.... 00000000
+01C8 DUA54      DUCB..... 08E2F000  FLAGS.... 00000000
+01D0 DUA55      DUCB..... 08E32000  FLAGS.... 00000000
+01D8 DUA56      DUCB..... 08E35000  FLAGS.... 00000000
+01E0 DUA57      DUCB..... 08E38000  FLAGS.... 00000000
+01E8 DUA58      DUCB..... 08E3B000  FLAGS.... 00000000
+01F0 DUA59      DUCB..... 08E3E000  FLAGS.... 00000000
+01F8 DUA60      DUCB..... 08E41000  FLAGS.... 00000000
+0200 DUA61      DUCB..... 08E44000  FLAGS.... 00000000
+0208 DUA62      DUCB..... 08E47000  FLAGS.... 00000000
+0210 DUA63      DUCB..... 08E4A000  FLAGS.... 00000000
+0218 DUA64      DUCB..... 08E4D000  FLAGS.... 00000000
+0220 DUA65      DUCB..... 08E50000  FLAGS.... 00000000
+0228 DUA66      DUCB..... 08E53000  FLAGS.... 00000000
+0230 DUA67      DUCB..... 08E56000  FLAGS.... 00000000
+0238 DUA68      DUCB..... 08E59000  FLAGS.... 00000000
+0240 DUA69      DUCB..... 08E5C000  FLAGS.... 00000000
+0248 DUA70      DUCB..... 08E5F000  FLAGS.... 00000000
+0250 DUA71      DUCB..... 08E62000  FLAGS.... 00000000
+0258 DUA72      DUCB..... 08E65000  FLAGS.... 00000000
+0260 DUA73      DUCB..... 08E68000  FLAGS.... 00000000
+0268 DUA74      DUCB..... 08E6B000  FLAGS.... 00000000
+0270 DUA75      DUCB..... 08E6E000  FLAGS.... 00000000
+0278 DUA76      DUCB..... 08E71000  FLAGS.... 00000000
+0280 DUA77      DUCB..... 08E74000  FLAGS.... 00000000
+0288 DUA78      DUCB..... 08E77000  FLAGS.... 00000000
+0290 DUA79      DUCB..... 08E7A000  FLAGS.... 00000000
+0298 DUA80      DUCB..... 08E7D000  FLAGS.... 00000000
+02A0 DUA81      DUCB..... 08E80000  FLAGS.... 00000000
Dispatchable Unit Control Block: DUCB0
EZBDUCB: 08D8D010
+0000 DUCB_EYE..... DUCB
+0004 DUCB_LENGTH..... 0100
+0006 DUCB_VERSION..... 0002
+0008 DUCB_TOKEN..... 08D8D010  0014001E 10000000 00000000
+0018 DUCB_DUSA..... 08D8D110
+001C DUCB_AVAIL_CHAIN..... 00000000
+0020 DUCB_DUAEP..... 08E83028
+0026 DUCB_ASID..... 001E
+0028 DUCB_ASCB..... 00FA4400
+002C DUCB_ATCB..... 007EC920
+0030 DUCB_ITCVT..... 08E853C8
+0034 DUCB_LOCKSHELD_COUNT..... 00000000
+0038 DUCB_LOCKS_TABLE..... 08D8D194
+003C DUCB_LOCKS_SUSPENDED..... 00000000
+0040 DUCB_LOCKS_SUSPENDED_NEXT..... 7FFAFAF1
+0044 DUCB_SUSPEND_TOKEN..... 00000000 40000000
+004C DUCB_JOBNAME..... TCPCS
+0054 DUCB_TSAB..... 098221F0
+0058 DUCB_TSEB..... 09822230
+005C DUCB_TSDB..... 08E85000
+0060 DUCB_TSDX..... 08E850C8
+0064 DUCB_TCP_ASCB..... 00FA4400
+0068 DUCB_STREAMHEAD..... 00000000

```

```

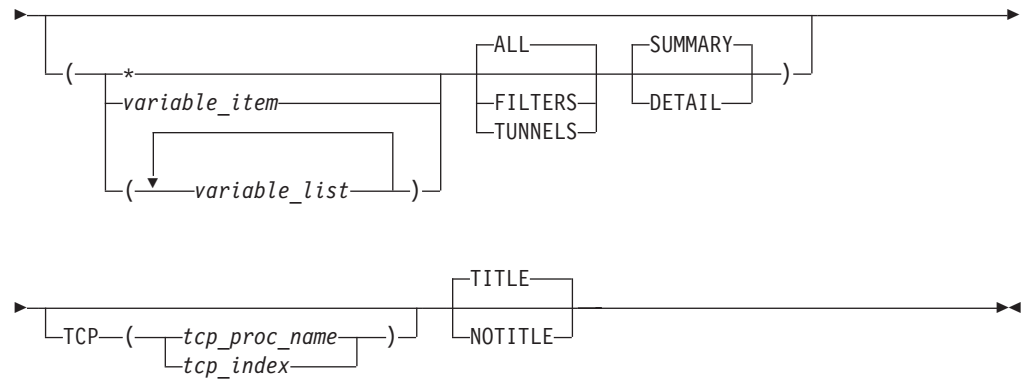
+006C DUCB_OSI..... 00000000
+0070 DUCB_CREATE_FLAGS..... 00000000
+0074 DUCB_CID..... 00000000
+0078 DUCB_PORT..... 0000
+007A DUCB_IPADDR_LEN..... 00
+007C DUCB_IPADDR..... 00010002 00030004 00050006 00070008
+008C DUCB_TRR_PTR..... 08D8D128
+0090 DUCB_ESTAEX_TOKEN..... 00000004
+0094 DUCB_ERRORCODE..... 00000000
+0098 DUCB_REASONCODE..... 00000000
+009C DUCB_ABEND_FLAGS..... 000F0000
+00A0 DUCB_DUMP_ECB..... 00000000
      DUCB_EXP_SAVE.....
+00B8 08D8D110 00000000 00000000 88E865C 08F671E8 7F4FC488 08D8F75C 08D8D010 08D90118 08D8F3A0 000001A8 08D8D010
+00E8 08D8F214 08E850C8 08D8FDA8 08D8FA30 0923E21C 88E8662C
Dispatchable Unit Stack Area: DUSA001
EZBDSA: 08D8D110
+0000 DUSA_EYE..... DUSA
+0004 DUSA_NEXTDUSA.. 08CFF010
+0008 DUSA_DUCB..... 08D8D010
+000C DUSA_START..... 08D8D450
+0010 DUSA_LAST..... 08D90000
+0014 DUSA_NEXTAVAIL. 08D8D540
Register Save Area: RSA001
Module: EZBTIWAT
EZBDSA: 08D8D458
+0000 RSA_DUSA. 08D8D110 RSA_PREV. 0A301890 RSA_NEXT. 00000000 RSA_R14.. FEFEFEFE RSA_R15.. FEFEFEFE RSA_R0... FEFEFEFE
+0018 RSA_R1... FEFEFEFE RSA_R2... FEFEFEFE RSA_R3... FEFEFEFE RSA_R4... FEFEFEFE RSA_R5... FEFEFEFE RSA_R6... FEFEFEFE
+0030 RSA_R7... FEFEFEFE RSA_R8... FEFEFEFE RSA_R9... FEFEFEFE RSA_R10.. FEFEFEFE RSA_R11.. FEFEFEFE RSA_R12.. FEFEFEFE
+0050 RSA_AR13. FEFEFEFE RSA_AR14. FEFEFEFE RSA_AR15. FEFEFEFE RSA_AR0.. FEFEFEFE RSA_AR1.. FEFEFEFE RSA_AR2.. FEFEFEFE
+0068 RSA_AR3.. FEFEFEFE RSA_AR4.. FEFEFEFE RSA_AR5.. FEFEFEFE RSA_AR6.. 08D773D0 RSA_AR7.. 7EFEFEFE RSA_AR8.. 08E85084
+0080 RSA_AR9.. 08E85238 RSA_AR10. 08E8523C RSA_AR11. 08E85350 RSA_AR12. 08E85358
Dynamic Area of RSA001
Module: EZBTIWAT
08D8D458 08D8D110 0A301890 00000000 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .QJ.....
+0020 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .....
+0040 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .....
+0060 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE 08D773D0 7EFEFEFE 08E85084 .....P.)=...Y&d
+0080 08E85238 08E8523C 08E85350 08E85358 88E85210 FEFEFEFE FEFEFEFE FEFEFEFE .Y...Y...Y.&Y..hY.....
+00A0 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .....
+00C0 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .....
+00E0 FEFEFEFE FEFEFEFE 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0100 00000000 00000000 00000000 00000000 00000000 00000000 08D8D4B8 00000000 .....QM.....
+0120 00FEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .....
+0140 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .....
+0160 00000000 0A301958 88F588A0 08D8D5B8 0A301890 08D8D010 08D8D128 09822230 .....h5h..QN.....Q}.QJ..b..
+0180 08D8D110 08F59E2D 08D8D5F8 08F57E2F 88E8662C 08D8D458 00000008 08D8D110 .QJ..5...QN8.5=.hYf..QM.....QJ.
+01A0 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .....
+01C0 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .....
+01E0 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE E3C3D7C3 E2404040 E2E3C3F0 F0F0F2F8 .....TCPCS STC00028
+0200 E9840A00 00000000 00FEFEFE FEFEFEFE 00000000 00000000 FEFEFEFE FEFEFEFE Zd.....
+0220 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE .....
+0240 7EFEFEFE 0000004F 0A301EA0 84080000 08F77BE4 0000004F 0A301EA4 00000008 =.....|....d....7#U...|...u....
+0260 08E85000 0A301EA0 08F59E2D 08F58E2E 08F7309B 88F7209C 08D8D458 88F721A4 .Y&.....5...5...7..h7...QM.h7.u
+0280 08D8D110 FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE FEFEFEFE 00030001 88F58830 .QJ.....h5h.
+02A0 00000004 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+02C0 00000002 00000000 08D8D4AC 00000004 00000000 08D8D738 00000100 FEFEFEFE .....QM.....QP.....
+02E0 003C8010 C5E9C1C9 D5F1F1C9 40C1D3D3 40E3C3D7 C9D740C9 D5E3C5D9 C6C1C3C5 ...EZAIN111 ALL TCPIP INTERFACE
+0300 E240C6D6 D940D7D9 D6C340E3 C3D7C3E2 40C1D9C5 40C1C3E3 C9E5C548 C400007C S FOR PROC TCPCS ARE ACTIVE....@
+0320 40000000 00000000 00000000 00000000 00100000 41000000 00000000 00000000 .....
+0340 00000000 00000000 E2E3C3F0 F0F0F2F8 E3C3D7C3 E2404040 40404040 40404040 .....STC00028TCPCS
+0360 00000000 00000001 40404040 40404040 40404040 00000000 00000000 00000000 .....
+0380 00000000 00FA4400 00000000 00000000 00000000 00000000 00000000 00000000 .....
+03A0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+03C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+03E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0400 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0420 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0440 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0460 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0480 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+04A0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+04C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+04E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0500 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0520 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0540 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0560 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0580 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+05A0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+05C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+05E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....
+0600 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 .....

```


+0620	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0640	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0660	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0680	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+06A0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+06C0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+06E0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0700	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0720	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0740	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0760	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0780	0000FEFE	FEFEFEFE	C5E9C2C3	E3D9C3C4	08D8D110	08D8D458	08D8DD88	FEFEFEFEEZBCTRC.D.QJ..QM..Q.h....
+07A0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+07C0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	88E88744	88E88AB8hYg.hY..
+07E0	00000000	FEFEFEFE	08D8D71C	08D8D6F0	08D8D010	FEFEFEFE	000AEF36	01FF000CQP..Q00.Q}.....
+0800	00000166	00000048	00000100	000AEF9A	08D8D738	08D8D4A0	FEFEFEFE	FEFEFEFEQP..QM.....
+0820	FEFEFEFE	FEFEFEFE	01660010	00030001	B64C8AD6	FD7CDBC4	001E001E	001E0004<.0.@.D.....
+0840	007EC920	88F58830	E3C3D7C3	E2404040	00000000	0000083A	00000000	00000000=I.h5h.TCPCS.....
+0860	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0880	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+08A0	00000000	00000000	08D8D4B8	00000000	00FEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFEQM.....
+08C0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+08E0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	00000000	08D8D4A0	88F58830	08D8DD48QM.h5h..Q..
+0900	08D8D010	08D8D010	08D8D128	08E85000	08D8D110	08D8D4AC	08D8DD88	08F7309BQ}..QJ..Y&..QJ..QM..Q.h.7..
+0920	88E8862C	08D8DBE8	00000008	08D8D110	00000000	00000000	00000000	00000000	hYf..Q.Y.....QJ.....
+0940	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0960	00000000	00000000	00000000	00000000	0000FEFE	FEFEFEFE	C5E9C2C3	E3D9C3C4EZBCTRC.D
+0980	08D8D110	08D8D648	08D8DF78	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	..QJ..Q0..Q.....
+09A0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+09C0	FEFEFEFE	FEFEFEFE	88E88744	88E88AB8	00000000	FEFEFEFE	08D8D90C	08D8D8E0hYg.hY.....Q.R..QQ..
+09E0	08D8D010	FEFEFEFE	00003004	01FF000C	00000166	00000048	00000100	00083068	..Q}.....
+0AA0	08D8D928	08D8D690	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	01660010	00030001	..Q.R..Q0.....
+0A20	B64C8AD3	98C96589	001E001E	001E0004	007EC920	88F61E4E	E3C3D7C3	E2404040	..<.LqI.i.....=I.h6.+TCPCS
+0A40	00000000	000005F5	00000000	00000000	00000000	00000000	00000000	000000005.....
+0A60	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0A80	00000000	00000000	00000000	00000000	00000000	00000000	012892A0	00000000k.....
+0AA0	00FEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0AC0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0AE0	00000000	08D8D690	88F61E4E	08D8DF38	08D8D010	08D8D010	08D8D128	08F5AE8CQ0.h6.+Q..Q}..QJ..5..
+0B00	08D8D110	08D8D69C	08D8DF78	08F7309B	88E8862C	08D8DD88	00000008	08D8D110	..QJ..Q0..Q..7..hYf..Q.Q.....QJ..
+0B20	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0B40	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0B60	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0B80	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0BA0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0BC0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0BE0	FEFEFEFE	FEFEFEFE	FEFEFEFE	08D8D010	08D8DD84	08D8DD88	08D8DDA4	08D8D604Q}..Q.d.Q.h.Q.u.Q0..
+0C00	08D8D600	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	..Q0.....
+0C20	00000000	08D8DF7C	08D8D010	00000000	08D8D9C4	FE000000	0A301E38	08D8E078QP..Q}.....QRD.....Q\..
+0C40	00000000	00000000	08D8E1F0	0903F30C	89035C6C	08D8D9B8	00000000	09035D20Q.0..3.i.*%.QR.....).
+0C60	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0C80	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0CA0	5001001F	890368CE	0000007E	0000000E	00000000	00000000	00000000	00000000	&...i.....=.....
+0CC0	00000000	00000000	00000004	00000000	08D8D5B0	00000004	00000000	08D8D5D8QN.....QNQ.....
+0CE0	00000020	00000000	08D8D5F8	00000010	00000000	08D8E170	00000040	FEFEFEFEQN8.....Q.....
+0D00	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	00010026	00000000
+0D20	00000000	00000000	00000000	00000028	00000000	00000000	00000000	00000000
+0D40	00000000	00000000	7F686108	00000000	00000000	00000000	FEFEFEFE	FEFEFEFE"/.....
+0D60	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0D80	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	C5E9C2C3	E3D9C3C4EZBCTRC.D
+0DA0	08D8D110	08D8D9B8	08D8E398	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	..QJ..QR..QTq.....
+0DC0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0DE0	FEFEFEFE	FEFEFEFE	88E88744	88E88AB8	00000000	FEFEFEFE	08D8E124	08D8E0F8hYg.hY.....Q..Q\8
+0EE0	08D8D010	FEFEFEFE	000378F6	01FF000C	000000EE	00000048	00000040	000379A2	..Q}.....6.....`s
+0E20	08D8E170	08D8DC70	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	00EE0010	5001001F	..Q..Q.....&...
+0E40	B64C8AC7	9AC2B089	001E001E	001E007E	007EC920	890368CE	E3C3D7C3	E2404040	..<.G.B.i.....=I.i..TCPCS
+0E60	00000000	000002F6	00000000	0000000E	00000000	00000000	00000000	000000006.....
+0E80	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0EA0	00000000	00000000	00000000	00000000	00000000	00000000	012892A0	00000000k.....
+0EC0	00FEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0EE0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0F00	00000000	08D8DC70	890368CE	08D8E358	08D8D9B8	08D8D010	08D8D128	08D8D010Q..i....QT..QR..Q}..QJ..Q}..
+0F20	08D8D110	00000010	08D8E398	0903F30C	88E8862C	08D8E1F8	00000008	08D8D110	..QJ.....QTq..3.hYf..Q.8.....QJ..
+0F40	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0F60	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0F80	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0FA0	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE
+0FC0	FEFEFEFE	FEFEFEFE	08D8E620	08D8E620	FEFEFEFE	FEFEFEFE	FEFEFEFE	7F207498QW..QW....."..q
+0FE0	09822230	FEFEFEFE	08E850C8	08D8D010	FEFEFEFE	FEFEFEFE	FEFEFEFE	FEFEFEFE	..b.....Y&H}.....

Dispatchable Unit Stack Area: DUSA002
EZBDUSA: 08CF010
+0000 DUSA_EYE..... DUSA
+0004 DUSA_NEXTDUSA.. 00000000

```
82 DUCBs were found
 2 DUCBs were formatted
Analysis of Tcp/Ip for TCPCS completed
```



Parameters

If no parameters are specified, all Firewall filters and tunnels are summarized. The following are the parameters for the TCPIP FIREWALL subcommand:

- * An asterisk is used as a placeholder if no variable parameters will be specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters may be repeated, separated by a blank space, within parentheses:

filter_address

Displays the Firewall filter with this address. An address is specified as one to eight hexadecimal digits. An IPCS symbol name may be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

tunnel_address

Displays the Firewall tunnel with this address. An address is specified as one to eight hexadecimal digits. An IPCS symbol name may be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

In addition to the variable parameters described above, the following keyword parameters may be specified:

ALL

Display information for all Firewall filters and tunnels. ALL is the default.

FILTERS

Only display information for Firewall filters.

TUNNELS

Only display information for Firewall tunnels.

SUMMARY

Displays the addresses of the control blocks and other data in tables. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL also shows the contents of the control blocks.

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {ALL, FILTERS, TUNNELS}, only the last one will be used.
2. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is a sample output of the TCIPCS FIREWALL subcommand.

```
TCIPCS FIREWALL ((* ) SUMMARY ALL )
Dataset: IPCS.A594094.DUMPN
Title:   FIREWALL DUMP
```

The address of the TSAB is: 08DE56F8

```
Tseb      SI Procedure Version Tsdb      TsdX      Asid Trace0pts Status
08DE5738  1 TCPCS      V2R10    0854B000 0854B0C8 01F6 9FFFFFFF Active
```

```
1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found

1 TCP/IP(s) for CS      V2R10 found
```

=====

Analysis of Tcp/Ip for TCPCS. Index: 1

TCPIP Firewall Analysis

Secure Adapters:

```
9.67.116.125
162.33.33.33
9.67.11.1
9.67.11.2
9.67.114.1
```

Pre-decap filtering : No

Filter Summary:

Action	Src1@	Src2@	Dst1@	Dst2@	SPort	DPort	Protocol
Permit	0.0.0.0	0.0.0.0	0.0.0.0	0.0.0.0	=500	=500	17 (UDP)
Permit	0.0.0.0	0.0.0.0	0.0.0.0	0.0.0.0	0	0	51 (SIPP-AH)
Permit	0.0.0.0	0.0.0.0	0.0.0.0	0.0.0.0	0	0	50 (SIPP-ESP)
Permit	0.0.0.0	0.0.0.0	9.67.113.36	255.255.255.255	0	=1014	6 (TCP)
Permit	9.67.113.36	255.255.255.255	0.0.0.0	0.0.0.0	=1014	0	6 (TCP)
Unknown	9.67.116.36	255.255.255.255	9.67.116.47	255.255.255.255	0	0	1 (ICMP)
Permit	9.67.116.36	255.255.255.255	9.67.116.47	255.255.255.255	0	0	1 (ICMP)
...							
Permit	9.67.116.47	255.255.255.255	9.67.113.4	255.255.255.255	0	0	254 (254)
Deny	0.0.0.0	0.0.0.0	0.0.0.0	0.0.0.0	0	0	254 (254)

Tunnel Summary:

Name	Src@	Dst@	Policy	Format
0000000510:0000000508:0000000507:0000000516:0000000517:0000000506:0000000003	9.67.116.36	9.67.116.47	000000021	000000033
0000000504:0000000508:0000000507:0000000504:0000000503:0000000506:0000000004	9.67.116.36	9.67.116.47	000000021	000000033
0000000510:0000000508:0000000507:0000000516:0000000518:0000000502:0000000005	9.67.116.36	9.67.116.47	000000042	0000000CC
0000000510:0000000508:0000000507:0000000516:0000000518:0000000502:0000000006	9.67.116.36	9.67.116.47	000000042	0000000CC

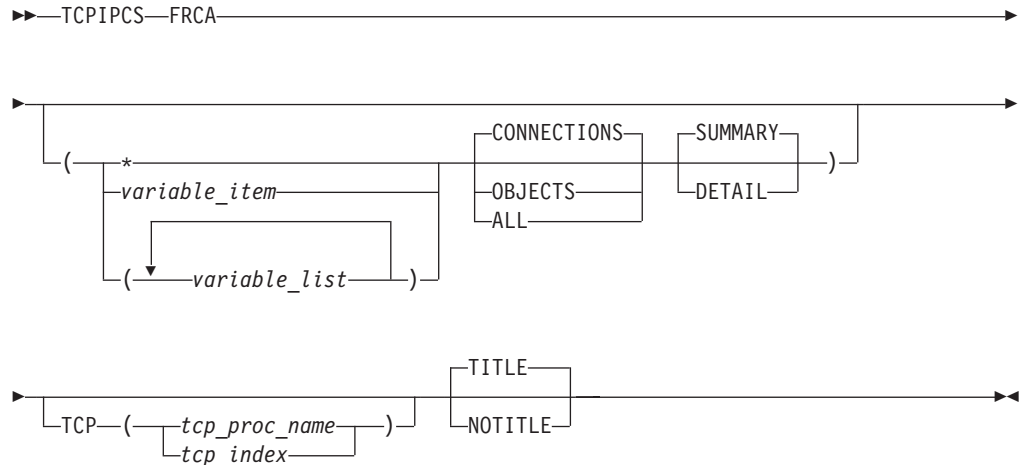
Analysis of Tcp/Ip for TCPCS completed

TCIPCS FRCA

Display information about the Fast Response Cache Accelerator (FRCA) connections or about cached objects.

Syntax

Following is the syntax of the TCIPCS FRCA subcommand:



Parameters

If no parameters are specified, only FRCA connections are summarized. The following are the parameters for the TCIPCS FRCA subcommand:

- * An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space, within parentheses:

TCB_address

Displays the FRCA connection with this address. An address is specified as one to eight hexadecimal digits. An IPCS symbol name may be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

UWSX_address

Displays the FRCA server connection with this address. An address is specified as one to eight hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

jobname

Displays only the FRCA information for this job name. The job name can be a TCP/IP application name or a stack name. The job name contains one to eight alphanumeric characters.

connection_id

Displays the FRCA information with this connection ID. An ID is specified as one to eight hexadecimal digits.

In addition to the variable parameters described above, the following keyword parameters can be specified:

CONNECTIONS

Only display information for FRCA connections. CONNECTIONS is the default.

OBJECTS

Only display information for FRCA cached objects.

ALL

Display information for all FRCA connections and cached objects.

SUMMARY

Displays the addresses of the control blocks and other data in tables.
SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL also shows the contents of the control blocks.

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {CONNECTIONS, OBJECTS, ALL}, only the last one will be used.
2. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIPCS FRCA subcommand:

```
TCPIPCS FRCA
Dataset: IPCS.MV20372.DUMPA
Title:   TCPSVT   V2R10: Job(TCPSVT  ) EZBITST0(HTCP50A 99.281)+
        00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051
```

The address of the TSAB is: 12E89BB8

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
12E89BF8	1	TCPSVT	V2R10	12B57000	12B570C8	0051	9FFFFFFF	Active

```
1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
```

```
1 TCP/IP(s) for CS      V2R10 found
```

=====

Analysis of Tcp/Ip for TCPSVT. Index: 1

FRCA Server Connections

Uwsx@	Tcb@	Cache@	References	Flags
12E6BA90	7F272D08	12E6AE98	10	60

FRCA Client Connections

Uwcx@	Tcb@	Server@	Object@	Flags
7F20E060	7F20DD08	12E6BA90	1299CB08	08
7F14D460	7F14D108	12E6BA90	1299BA88	48
7F1FAC60	7F1FA908	12E6BA90	12434488	48
7F4DD460	7F4DD108	12E6BA90	00000000	28
7F0A9060	7F0A8D08	12E6BA90	00000000	28

```
7F08F460 7F08F108 12E6BA90 00000000 28
7F066860 7F066508 12E6BA90 00000000 00
```

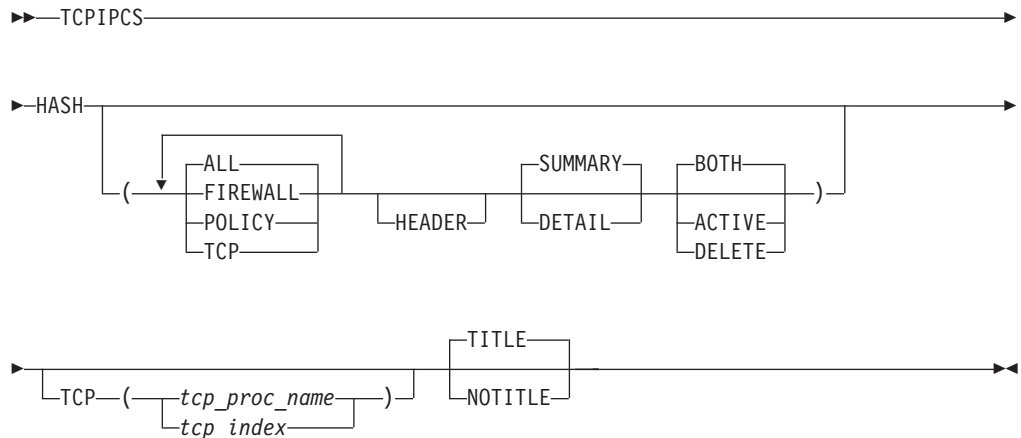
Analysis of Tcp/Ip for TCPSVT completed

TCIPCS HASH

Display information about the structure of TCP/IP hash tables.

Syntax

Following is the syntax of the TCIPCS HASH subcommand:



Parameters

The following keyword parameters may be specified:

ALL

Display structure of all TCP/IP hash tables. ALL is the default.

FIREWALL

Only display structure of Firewall hash tables.

ICMPV6

Only display structure of ICMPV6 hash tables.

POLICY

Only display structure of Service Policy hash tables.

TCP

Only display structure of TCP hash tables.

UDP

Only display structure of UDP hash tables.

HEADER

Display hash table header information. Not displayed by default.

SUMMARY

Displays the addresses of the control blocks and other data in tables. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL also shows the search key values.

BOTH

Display both active and logically deleted table elements. BOTH is the default.

ACTIVE

Only display active table elements.

DELETE

Only display logically deleted table elements.

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {ALL, FIREWALL, ICMPV6, POLICY, TCP, UDP}, all of them will be used.
2. If you specify multiple keywords from the set {BOTH, ACTIVE, DELETE}, only the last one will be used.
3. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIPCS HASH subcommand.

```
TCPIPCS HASH ( DETAIL ALL )
Dataset: D74L.KWDEV03A.DUMP
Title: ICMP HASHTAB
The address of the TSAB is: 0999D6F8
Tseb      SI Procedure Version TsdB      TsdX      Asid TraceOpts Status
0999D738  1 TCPCS2      V1R4      08FE9000 08FE90C8 0013 00000000 Active
0999D7B8  2 TCPCS      V1R4      08FB2000 08FB20C8 002D 00000000 Active
      2 defined TCP/IP(s) were found
      2 active TCP/IP(s) were found
      2 TCP/IP(s) for CS      V1R4      found
=====
Analysis of Tcp/Ip for TCPCS2. Index: 1
TCPIP Hash Table Analysis
Policy ID Port Table
Hash Table Header at 7F65E008
Instance          : 1
Active entries     : 0
Hash buckets       : 1,999
User free routine  : 00000000
Element queue      : 08FE9E48
0 elements in Policy Id Port Table
Table Summary:
Active buckets     : 0
Inactive buckets   : 0
Unused buckets     : 1,999
Max active q length : 0
Max active q index : 0
Max active q seqnum : 0
Max delete q length : 0
Max delete q index : 0
Total seqnum       : 0

ICMPV6 Table
Hash Table Header at 7F699C08
Instance          : 4
Active entries     : 7
Hash buckets       : 1,024
User free routine  : 00000000
Element queue      : 08FE9E50
Bucket# Bucket@ Element@ Status User@ KeyValue
      2 7F699C28 7F2F8E80 Active 0AA6BFD8 FEC00000 00000000 00000000 00000000
Clock Ticks..... 00000003
Tokens..... 00
Token Tenths..... 00
```



```

5 7F699C58 7F2F9100 Active 0AA6BF98 00000000 00000000 00000000 00000000
Clock Ticks..... 00000008
Tokens..... 00
Token Tenths..... 00
6 7F699C68 7F2F9080 Active 0AA6BFA8 00000000 00000000 00000000 00000000
Clock Ticks..... 00000011
Tokens..... 00
Token Tenths..... 00

```

7 elements in ICMPV6 Table

Table Summary:

```

Active buckets      : 6
Inactive buckets    : 0
Unused buckets      : 1,018
Max active q length : 2
Max active q index  : 6
Max active q seqnum : 2
Max delete q length : 0
Max delete q index  : 0
Total seqnum        : 7

```

TCP V4 Index Table

Hash Table Header at 7F528B88

```

Instance      : 2
Active entries : 6
Hash buckets   : 62,533
User free routine : 88D9523E
Element queue  : 08FE9E48

```

Bucket#	Bucket@	Element@	Status	User@	KeyValue
0	7F528B88	7F507FE0	Active	7F510108	00000000 00000000 00000000 00000000
530	7F52ACA8	7F5080C0	Active	7F51B988	00000000 00000000 00150000
30479	7F59FC78	7F508020	Active	7F510A08	7F000001 00000000 04000000
35181	7F5B2258	7F5080A0	Active	7F51A788	00000000 00000000 27170000
37771	7F5BC438	7F508040	Active	7F511308	7F000001 7F000001 04000401
40773	7F5C7FD8	7F508060	Active	7F510E88	7F000001 7F000001 04010400

6 elements in TCB V4 Index Table

Table Summary:

```

Active buckets      : 6
Inactive buckets    : 1
Unused buckets      : 62,526
Max active q length : 1
Max active q index  : 0
Max active q seqnum : 1
Max delete q length : 0
Max delete q index  : 0
Total seqnum        : 8

```

TCP V6 Index Table

Hash Table Header at 7F2FDB88

```

Instance      : 5
Active entries : 2
Hash buckets   : 62,533
User free routine : 88D9523E
Element queue  : 08FE9E50

```

Bucket#	Bucket@	Element@	Status	User@	KeyValue
0	7F2FDB88	7F2F8C80	Active	7F510588	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
530	7F2FFCA8	7F2F8F00	Active	7F51B988	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

2 elements in TCB V6 Index Table

Table Summary:

```

Active buckets      : 2
Inactive buckets    : 0
Unused buckets      : 62,531
Max active q length : 1

```

```

Max active q index : 0
Max active q seqnum : 1
Max delete q length : 0
Max delete q index : 0
Total seqnum : 2

UDP DMUX V4 Table
Hash Table Header at 7F403B88
Instance : 3
Active entries : 2
Hash buckets : 62,533
User free routine : 88DB0E3C
Element queue : 08FE9E48
Bucket# Bucket@ Element@ Status User@ KeyValue
0 7F403B88 7F508000 Active 7F4F8108 00000000 0000
529 7F405C98 7F508080 Active 7F500608 00000000 0211

```

2 elements in UDP DMUX V4 Table

```

Table Summary:
Active buckets : 2
Inactive buckets : 0
Unused buckets : 62,531
Max active q length : 1
Max active q index : 0
Max active q seqnum : 1
Max delete q length : 0
Max delete q index : 0
Total seqnum : 2

```

```

UDP DMUX V6 Table
Hash Table Header at 7F203B88
Instance : 6
Active entries : 1
Hash buckets : 62,533
User free routine : 88DB0E3C
Element queue : 08FE9E50
Bucket# Bucket@ Element@ Status User@ KeyValue
0 7F203B88 7F2F8D00 Active 7F4F8808 00000000 00000000 00000000 00000000
0000

```

1 elements in UDP DMUX V6 Table

```

Table Summary:
Active buckets : 1
Inactive buckets : 0
Unused buckets : 62,532
Max active q length : 1
Max active q index : 0
Max active q seqnum : 1
Max delete q length : 0
Max delete q index : 0
Total seqnum : 1

```

```

UDP MULTICAST V6 Table
Hash Table Header at 7F10EB88
Instance : 7
Active entries : 0
Hash buckets : 62,533
User free routine : 88DB0E3C
Element queue : 08FE9E50
0 elements in UDP MULTICAST V6 Table
Table Summary:
Active buckets : 0
Inactive buckets : 0
Unused buckets : 62,533
Max active q length : 0
Max active q index : 0
Max active q seqnum : 0

```

```

|           Max delete q length : 0
|           Max delete q index  : 0
|           Total seqnum         : 0
|           Analysis of Tcp/Ip for TCPCS2 completed

```

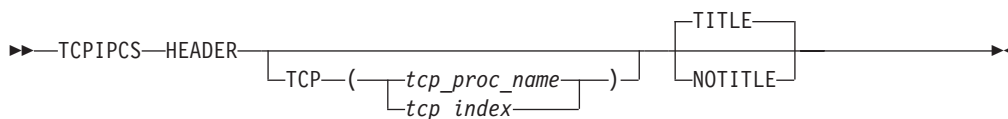
TCPIPCS HEADER

The TCPIPCS HEADER command displays information from the system dump header and, in some cases, if a DUCB has ABENDED, the DUCB is displayed. The IPCS command "STATUS System Cpu Registers Worksheet Faildata" is used to display the system dump header.

Depending on the error recovery routine, the DUCB address may or may not be available. If the DUCB address is available, the DUCB will be displayed. To find DUCBs that ABENDED, use the TCPIPCS DUAF (* ABEND) command.

Syntax

Following is the syntax of the TCPIPCS HEADER subcommand:



Parameters

The following keyword parameters may be specified:

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Sample output

The following is sample output of the TCPIPCS HEADER subcommand:

```

TCPIPCS HEADER
Dataset: IPCS.MV21381.DUMPA
Title:   SLIP DUMP ID=TC

```

The address of the TSAB is: 13391BC0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
13391C00	1	TCPSVT	V2R10	1323B000	1323B0C8	07DE	04041405	Active
13391C80	2	TCPSVT2	V2R10	00000000	00000000	07E8	00000000	Down Stopping
13391D00	3	TCPSVT1	V2R10	12FC3000	12FC30C8	0080	94FF755F	Active
13391D80	4	TCPSVT3	V2R10	00000000	00000000	0059	00000000	Down Stopping

```

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

```

```

4 TCP/IP(s) for CS      V2R10 found

```

```

=====

```

Analysis of Tcp/Ip for TCPSVT. Index: 1

STATUS SUBCOMMAND

MVS Diagnostic Worksheet

Dump Title: SLIP DUMP ID=TC

CPU Model 9672 Version AC Serial no. 041018 Address 00
Date: 03/22/2000 Time: 07:36:57.297123 Local

Original dump dataset: SYS1.DUMP93

Information at time of entry to SVCDUMP:

HASID 000B PASID 000B SASID 000B PSW 440C0000 81584B1C

CML ASCB address 00000000 Trace Table Control Header address 7F45D000

Dump ID: 007

Error ID: N/A

SDWA address N/A

....

CPU STATUS:

PSW=440C0000 81584B1C (RUNNING IN PRIMARY, KEY 0, AMODE 31, DAT ON)
DISABLED FOR I/O EXT

ASID(X'000B') 01584B1C. IEANUC09.IEAVEDS0+1C IN READ ONLY NUCLEUS

ASCB11 at FBD700, JOB(WLM), for the home ASID

ASXB11 at 7FDFA0 and TCB11M at 7FB440 for the home ASID

HOME ASID: 000B PRIMARY ASID: 000B SECONDARY ASID: 000B

GPR VALUES

0-3 00000001 0288E01C 00000C38 00000008

4-7 007FB440 007FFC10 007F6A68 00FBD700

8-11 00000000 01584B00 015AD820 007FEE48

12-15 00000000 00000000 80FDE336 81584B18

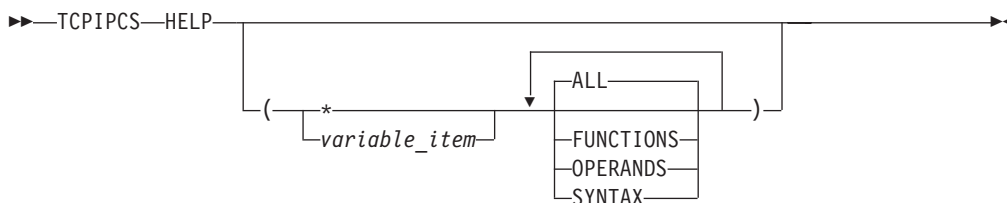
...

TCPIPCS HELP

Display TCPIPCS usage and syntax information.

Syntax

Following is the syntax of the TCPIPCS HELP subcommand:



Parameters

If no parameters are specified, the function, operand, and syntax information are displayed for all TCPIPCS commands.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the TCPIPCS subcommand names.

In addition to the variable parameters described above, the following keyword parameters can be specified:

ALL

Display information for all TCPIPCS commands. ALL is the default.

FUNCTION

Only display function information.

OPERANDS

Only display operand information.

SYNTAX

Only display syntax information.

Note: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them will be used.

Sample output

The following is sample output of the TCPIPES HELP subcommand:

```
tcpipecs help (config function)
```

Function:

The TCPIPES command displays selected information about a specific TCP/IP address space.

CONFIG - Produce device configuration report.

Function:

Display information about device, physical, and logical interfaces

Syntax:

```
TCPIPES CONFIG(<{SUMMARY|DETAIL}>)
```

Operands:

SUMMARY - Display summary report.

DETAIL - Display summary and interface cross-reference reports.

TCPIPES LOCK

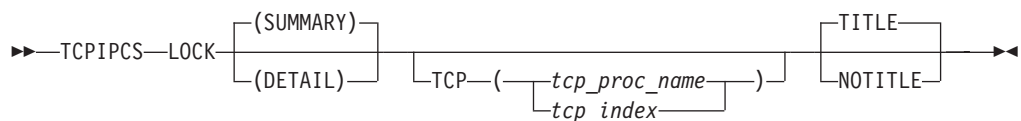
The TCPIPES LOCK command scans the dump for information about the current locks that are defined and held.

Only nonzero statistics are reported.

Note: The DUCB lock table entries may conflict with the lockword counters. This is because DUCB lock table entries and lockword counters are not updated in one operation, therefore they can be out of sync. At the moment the dump was obtained, the lockword counters may have been updated but the DUCB has not yet been updated.

Syntax

Following is the syntax of the TCPIPES LOCK subcommand:



Parameters

Following are the parameters for the TCPIPCS LOCK subcommand:

SUMMARY

Displays each level of each class of lock, the total number of DUCBs found, and a cross-reference for each lock being used. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL also shows lock information for each DUCB.

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIPCS LOCK subcommand:

```
TCPIPCS LOCK (DETAIL)
Dataset: IPCS.A594094.DUMPM
Title:  TCPSVT  V3R10: Job(TCPSVT  ) EZBITST0(HTCP50A 99.281)+
       00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051
...
ItCvt: 12B573C8, Class_Count: 12, Level_Count: 34, Table_Size: 616

Lock statistics at 12E7B208

  Class 2 at 12E7B2E8 for 2 levels
    Level 0201 ITSTOR_QUE
      Suspension - Srb :          1,601
      Delays      -      :          239
    ...
  Class 6 at 12E7B478 for 4 levels
    Level 0602 TCB
      Suspension - Srb :          146
      Suspension - Tcb :           33
    ...

Ix  Duct@  Lktb@  Susp@  Next@  DuctIx  Status
0002 12A62000 12A62184 00000000 00000000 10000001 Iu
      Lock Class 02: 00000001 00000002 12A62278 00000000
      Lock Level 01: 12B57CB8 C0010201 00010000 Held Excl      ITSTOR_QUE

Ix  Duct@  Lktb@  Susp@  Next@  DuctIx  Status
072E 12B19000 12B19184 00000000 7FFAFAF1 1000003E Iu
      Lock Class 06: 00000002 00000004 12B192F0 00000000
      Lock Level 02: 7F272D38 80010602 00020100 Held Shr      TCB

50 DUCBs found
2 DUCBs held locks
0 DUCBs were waiting for locks

Lockword Cross Reference

Lock@  Duct@  Status      Name
12B57CB8      Not Held    ITSTOR_QUE
7F272D38 12B19000 Held Shr      TCB

2 locks were referenced

Lock Class/Level Multiple Usage:

Class Level Names
  03      02 REASM
```

```

PTREE
MGRP
...
0C 06 SKITSSL
TCFG_CLEANUP

```

Analysis of Tcp/Ip for TCPSVT completed

TCIPCS MAP

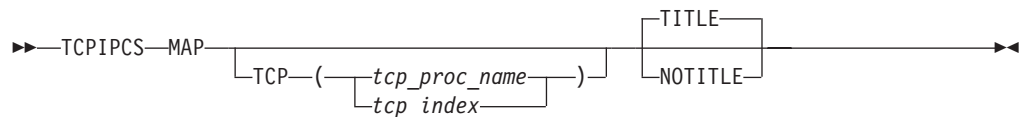
Invocation of this command displays a mapping of TCP/IP storage. This command is useful for finding overlays and abandoned storage.

Each control block referenced is listed in order by its address. Each control block eye-catcher is shown; if none is found, a mnemonic name is given in quotation marks. The size is the number of bytes (in decimal) in the control block. The key is the storage key. The base and offset are the address of a TCP/IP control block and the offset within it that contains the CbAddr in the far left column. There may be multiple references, so additional references are continued on a separate line.

Note: Large dumps with many control blocks can take considerable time to process.

Syntax

Following is the syntax of the TCIPCS MAP subcommand:



Parameters

Following are the parameters for the TCIPCS MAP subcommand:

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Sample output

The following is sample output of the TCIPCS MAP subcommand:

```

TCIPCS MAP
Dataset: IPCS.MV20767.DUMPA
Title:  VERIFY MV20758

```

The address of the TSAB is: 08DD36F8

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	Trace	Opts	Status
08DD3738	1	TCPCS	V2R10	0876E000	0876E0C8	01F7	92208100		Active

```

1 defined TCP/IP(s) were found
1 active  TCP/IP(s) were found

```

```

1 TCP/IP(s) for CS      V2R10 found

```

=====

Analysis of Tcp/Ip for TCPCS. Index: 1

CbIds enclosed in quotes e.g. "CBID" are not true eyecatchers.

Found 847 References and 1037 Cross-references

CbAddr	CbId	Size	Key	Base	+Offset
00FCC6A0	CVT	1,280	6		
01663450	ECVT	576	6	00FCC6A0+008C	
0876B000	"ALCCSA"	96	6	08DD9328+0004	
				08DD9368+0000	
0876B388	"CACSM"	120	6	0876B408+0004	
0876B408	"CACSM"	120	6	0876E5C8+0560	
0876B488	"CACSM"	120	6	0876B688+0004	
0876B500	"ACSA "	120	6	0876B600+000C	
0876B580	"ACSA "	120	6	0876B500+000C	
0876B600	"ACSA "	120	6	0876E5C8+0218	
0876B688	"CACSM"	120	6	0876E5C8+0568	
0876B700	"ACSA "	120	6	0876D700+000C	
0876B780	"ACSA "	120	6	0876B700+000C	
...					
7F6E8B78	SKQU	64	6	7F6E8748+00E8	
7F6E8BB8	SKQU	64	6	7F6E8AA8+0004	
7F6E8BF8	SKQP	16	6	7F6E8B78+0018	
				7F6E8BB8+0018	
7F6E8C08	SKBD	32	6	7F6E8B78+002C	
7F6E8C28	SKBD	32	6	7F6E8C08+0004	
7F6E8C48	SKBD	32	6	7F6E8BB8+002C	
7F6E8C68	SKBD	32	6	7F6E8C48+0004	
7F6E8CC8	SKSC	176	6	7F6E8008+0004	
				7F6E8748+0060	
7F6E8D88	SKRT	128	6	0876E0C8+0130	

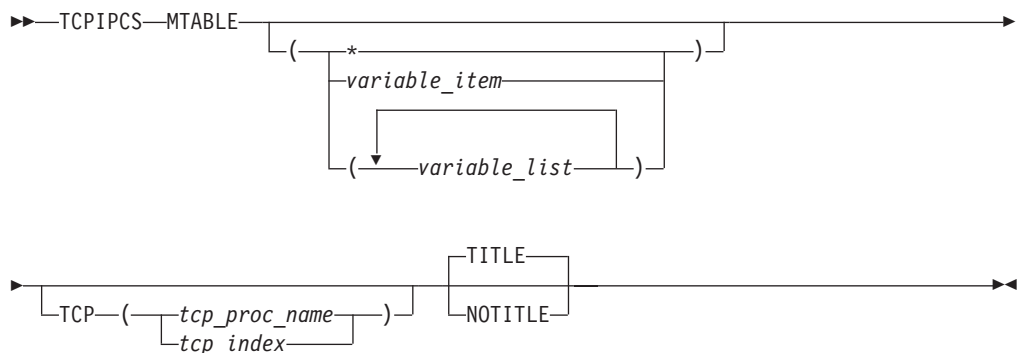
Analysis of Tcp/Ip for TCPCS completed

TCIPCS MTABLE

Invocation of this subcommand accesses the module tables and displays the module entry point address, name, compile date and time, PTF number, and load module name. The entries are listed first in entry-point-address order and then listed again in module-name order.

Syntax

Following is the syntax of the TCIPCS MTABLE subcommand:



Parameters

Following are the parameters for the TCIPCS MTABLE subcommand. If no parameters are specified, all displayable modules are displayed.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1-32 of the following variable parameters may be repeated, each separated by a blank space, within parentheses:

address

Locates the TCP/IP module where this address appears and displays the name and offset. An address is specified as one to eight hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

name Locates the TCP/IP module with this name. A name is specified as one to eight characters.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Sample output

The following is a sample output of the TCPIPCS MTABLE subcommand:

```
TCPIPCS MTABLE (12DE3800 12D9B858)
Dataset: IPCS.A594094.DUMPM
Title:   TCPSVT   V2R10: Job(TCPSVT ) EZBITST0(HTCP50A 99.281)+
        00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051
```

The address of the TSAB is: 12E89BB8

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
12E89BF8	1	TCPSVT	V2R10	12B57000	12B570C8	0051	9FFFFFFF	Active

```
1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
```

```
1 TCP/IP(s) for CS      V2R10 found
```

=====

Analysis of Tcp/Ip for TCPSVT. Index: 1

TCPIP Module Table Analysis

```
TCMT 12B590E8 EZBITCOM Size: 00D8 Cnt: 47
MTBL 12C23F28 EZBTIINI Size: 0CD4 Cnt: 272
MTBL 948ACA50 EZBTZMST Size: 0134 Cnt: 24
MTBL 94FE8470 EZBTTMST Size: 0704 Cnt: 148
MTBL 94AA0B00 EZBTMCTL Size: 0380 Cnt: 73
```

Module	Epa	Date	Time	PTF	Lmod	Asid
EZBIFARP	12DE35D8	1999/10/15	07:01:58	HTCP50A	EZBTIINI	0051
EZBXFINI	12D9B808	1999/10/08	00:37:29	HTCP50A	EZBTIINI	0051

```
Address 12DE3800 is EZBIFARP+0228
Address 12D9B858 is EZBXFINI+0050
```

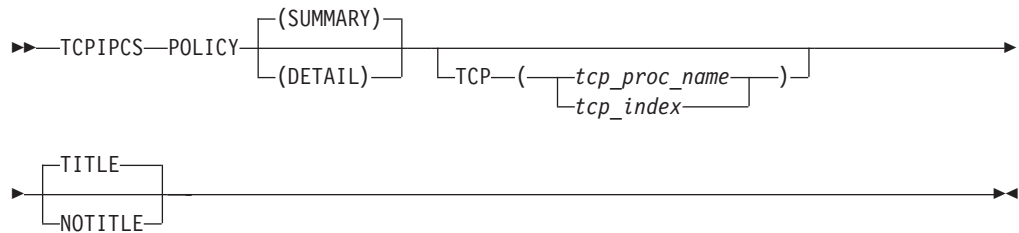
Analysis of Tcp/Ip for TCPSVT completed

TCPIPCS POLICY

Display information about policies.

Syntax

Following is the syntax of the TCIPICS POLICY subcommand:



Parameters

Following are the parameters for the TCIPICS POLICY subcommand:

SUMMARY

Displays the policy table addresses. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL also shows control block contents.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCIPICS POLICY subcommand:

```

TCIPICS POLICY TCP(1)
Dataset: IPCS.MV21046.DUMPA
Title:   BOTSWANA HUNG RUNNING PAGENT DIFFSERV SETTINGS.
  
```

The address of the TSAB is: 12EFD818

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
12EFD858	1	TCPSVT	V2R10	12EAB000	12EAB0C8	0058	9CFF755F	Active
12EFD8D8	2	TCPSVT1	V2R10	12A0F000	12A0F0C8	0069	9CFF755F	Active
12EFD958	3	TCPSVT2	V2R10	127C9000	127C90C8	07DE	9CFF755F	Active
12EFD9D8	4	TCPSVT3	V2R10	126FB000	126FB0C8	0054	9CFF755F	Active
12EFDA58	5	TCPSVT4	V2R10	12646000	126460C8	004C	9CFF755F	Active
12EFDAD8	6	TCPSVT5	V2R10	1260E000	1260E0C8	07DD	9CFF755F	Active
12EFDB58	7	TCPSVT6	V2R10	12383000	123830C8	007A	9CFF755F	Active
12EFDBD8	8	TCPSVT7	V2R10	11ECE000	11ECE0C8	07DC	9CFF755F	Active

8 defined TCP/IP(s) were found

8 active TCP/IP(s) were found

8 TCP/IP(s) for CS V2R10 found

=====
Analysis of Tcp/Ip for TCPSVT. Index: 1

Policy Control Table at 12F54210

Intrusion Detection Main Table at 13AA6088

Service Classes:

```

Scentry@ Scope Tos Pri Permission Name
129455F0 Both 60 00 Allowed paPRD-GenImp5
129454F0 Both 00 00 Allowed padefault
129453F0 Both E0 00 Allowed pa0SPF-1
129452F0 Both E0 00 Allowed paTST-1-GenImp1
129451F0 Both C0 00 Allowed paTST-1-GenImp2
12942B10 Both A0 00 Allowed paTST-1-GenImp3
12942A10 Both 80 00 Allowed paTST-1-GenImp4
12942910 Both 60 00 Allowed paTST-1-GenImp5
12942810 Both 40 00 Allowed paTST-1-GenImp6
12942710 Both 20 00 Allowed paTST-1-GenImp7
...

Policy Rules:

Preentry@ Permission Cond Level@ Cond@ Name
126C04F0 Allowed DNF 00000000 00000000 prPRD-CBS-30001
128F2A90 Allowed CNF 126EB590 00000000 prTST-WEB-4-80-B0
126C0B10 00000000
126C0790 126C0950
...

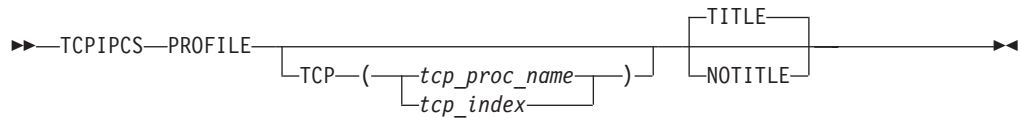
```

TCPIPCS PROFILE

Invocation of this subcommand shows the active configuration information at the time of the dump, in the form of profile data set statements. This profile does not necessarily match the profile used to start TCP/IP because the startup profile might not include the dynamic changes, additions, or deletions made using commands. All the defaults that are in effect are displayed in addition to explicit settings.

Syntax

Following is the syntax of the TCPIPCS PROFILE subcommand:



Parameters

Following are the parameters for the TCPIPCS PROFILE subcommand:

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Sample output

The following is sample output of the TCPIPCS PROFILE subcommand:

```

TCPIPCS PROFILE
Dataset: IPCS.R450697.V6TCBD1
Title: TCPCS2 CLIENT SIDE

```

The address of the TSAB is: 09DBE1A0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
09DBE1E0	1	TCPCS	V1R4	096C4000	096C40C8	0033	10841004	Active
09DBE260	2	TCPCS2	V1R4	096C9000	096C90C8	0034	10841004	Active

```

2 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

```

```

2 TCP/IP(s) for CS V1R4 found

```

=====
Analysis of Tcp/Ip for TCPCS2. Index: 2

```
;
; Profile generated on 2002/01/24 at 19:40:20
;
; Dump Dataset : IPCS.R450697.V6TCBD1
; Dump Time : 2001/11/07 17:41:23.649345
; TCP/IP Jobname: TCPCS2
;
;
; For informational purposes, only BEGINRoutes will
; be generated in this reconstructed profile.
;
; Either GATEWAY or BEGINRoutes statements may be
; specified in a real profile/obeyfile dataset.
; BEGINRoutes statement is the recommended way
; to define static routes.
;
```

ARPAGE 20

BEGINRoutes

```
ROUTE 9.67.115.83 HOST = OSAQDIOL MTU 8992 MAXImumretransmittime 120
MINImumretransmittime 0.5 ROUNDTRIPGain 0.125 VARIANCEGain 0.25
VARIANCEMultiplier 2 DELAYAckS NOREPLaceable
ROUTE 9.67.115.79 HOST = OSAQDIOL MTU 1500 MAXImumretransmittime 120
MINImumretransmittime 0.5 ROUNDTRIPGain 0.125 VARIANCEGain 0.25
VARIANCEMultiplier 2 DELAYAckS NOREPLaceable
ROUTE 9.67.115.0 255.255.255.192 = OSAQDIOL MTU 8992
MAXImumretransmittime 120 MINImumretransmittime 0.5
ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
DELAYAckS NOREPLaceable
ROUTE DEFAULT 9.67.115.1 OSAQDIOL MTU 8992 MAXImumretransmittime 120
MINImumretransmittime 0.5 ROUNDTRIPGain 0.125 VARIANCEGain 0.25
VARIANCEMultiplier 2 DELAYAckS NOREPLaceable
ROUTE FEC9:C2D4:1::9:67:115:83/128 FEC9:C2D4:1::206:2AFF:FE66:C81C
OSAQDI26 MTU 8992 MAXImumretransmittime 120
MINImumretransmittime 0.5 ROUNDTRIPGain 0.125 VARIANCEGain 0.25
VARIANCEMultiplier 2 DELAYAckS NOREPLaceable
ROUTE FEC9:C2D4:1::9:67:115:79/128 = OSAQDI26 MTU 1500
MAXImumretransmittime 120 MINImumretransmittime 0.5
ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
DELAYAckS NOREPLaceable
ROUTE FEC9:C2D4:1::206:2AFF:FE66:C81C/128 = OSAQDI26 MTU 8992
MAXImumretransmittime 120 MINImumretransmittime 0.5
ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
DELAYAckS NOREPLaceable
ROUTE 50C9:C2D4:1::206:2AFF:FE66:C81C/128 = OSAQDI26 MTU 8992
MAXImumretransmittime 120 MINImumretransmittime 0.5
ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
DELAYAckS NOREPLaceable
ROUTE 50C9:C2D4:1::0/112 50C9:C2D4:1::206:2AFF:FE66:C81C OSAQDI26 MTU
8992 MAXImumretransmittime 120 MINImumretransmittime 0.5
ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
DELAYAckS NOREPLaceable
ROUTE FEC9:C2D4:1::0/112 FEC9:C2D4:1::206:2AFF:FE66:C81C OSAQDI26 MTU
8992 MAXImumretransmittime 120 MINImumretransmittime 0.5
ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
DELAYAckS NOREPLaceable
ROUTE DEFAULT FEC9:C2D4:1::206:2AFF:FE66:C81C OSAQDI26 MTU 8992
MAXImumretransmittime 120 MINImumretransmittime 0.5
ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
DELAYAckS NOREPLaceable
```

```

ENDRoutes

DEvIce OSAQDI03 MPCIPA SECRouTER NOAUTORESTART
LiNK OSAQDIOL IPAQENET OSAQDI03 IFSPEED 1000000000

GLOBALCONFiG NOTCPStatistics ECSALIMIT 0K POOLLIMIT 0K

HOME
  9.67.115.82 OSAQDIOL

INTERFace OSAQDI26 DEFINE IPAQENET6 PORTNAME OSAQDI03 SECRouTER
  DUPADDRDET 1
  IPADDR FEC9:C2D4:1::9:67:115:82

IPCONFiG6 DATAGRamfwd(NOFWDMULTipath) NOSOURCEVIPA NOMULTIPATH HOPLimit
  255 ICMPErrorlimit 1000 NOIGNOREROUTERHOPLIMIT

IPCONFiG ARPTO 1200 DATAGRamfwd(NOFWDMULTipath) NOSOURCEVIPA
  NOTCPSTACKSOURCEVIPA NOVARSUBNETTING NOSYSPLExRouting
  REASSEMBlytimeout 60 TTL 64 NOPATHMTUDIScovery NOMULTIPATH
  NODYNAMICXCF NOIQDIORouting FORMAT(LONG)

ITRACE OFF AUTODAEMON
ITRACE OFF COMMAND
ITRACE OFF CONFiG
ITRACE OFF SUBAGENT

PKTTRACE FULL LINKNAME=LOOPBACK PROT=* IP=* SRCPort=* DESTport=*
PKTTRACE ON LINKNAME=LOOPBACK
PKTTRACE FULL LINKNAME=OSAQDIOL PROT=* IP=* SRCPort=* DESTport=*
PKTTRACE ON LINKNAME=OSAQDIOL

PRImaryinterface OSAQDIOL

SACONFiG COMMUnity public AGENT 161 ENABLED SETSDISAbled

SMFCONFiG TYPE118 NOTCPINIT NOTCPTERM NOFTPCLIENT NOTN3270CLIENT
  NOTCPIPStatistics TYPE119 NOTCPINIT NOTCPTERM NOFTPCLIENT
  NOTN3270CLIENT NOTCPIPSTATISTICS NOIFSTATISTICS
  NOPORTSTATISTICS NOTCPSTACK NOUDPTErm

SMFPARMS 0 0 0

SOMAXCONN 10

START OSAQDI03
START OSAQDI26

TCPCONFiG INTerval 120 RESTRICTLowports TCPRCVBufrsize 16384
  TCPSENDBufrsize 16384 TCPMAXRCVBufrsize 262144 FINWAIT2TIME
  600 SENDGarbage FALSE TCPTIMESTAMP

UDPCONFiG RESTRICTLowports UDPCHKsum UDPRCVBufrsize 65535
  UDPSENDBufrsize 65535 UDPQueuelimit

Analysis of Tcp/Ip for TCPCS2 completed

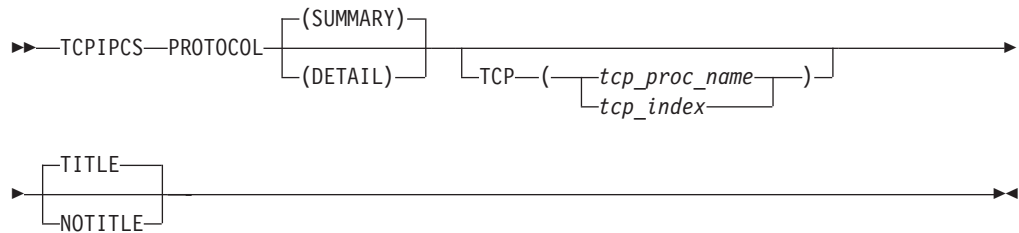
```

TCPIPCS PROTOCOL

Display information from TCP, UDP, and raw protocol control blocks.

Syntax

Following is the syntax of the TCPIPCS PROTOCOL subcommand:



Parameters

Following are the parameters for the TCPIPCS PROTOCOL subcommand:

SUMMARY

Formats the MTCB, MUDP, and MRCB contents. Lists all the TCBs, UCBs, and RCBs in separate cross-referenced tables. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL formats the contents of the TCBs, UCBs, and RCBs.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIPCS PROTOCOL subcommand:

```

TCPIPCS PROTOCOL
Dataset: IPCS.MV21381.DUMPA
Title:   SLIP DUMP ID=TC

```

The address of the TSAB is: 13391BC0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
13391C00	1	TCPSVT	V2R10	1323B000	1323B0C8	07DE	04041405	Active
13391C80	2	TCPSVT2	V2R10	00000000	00000000	07E8	00000000	Down Stopping
13391D00	3	TCPSVT1	V2R10	12FC3000	12FC30C8	0080	94FF755F	Active
13391D80	4	TCPSVT3	V2R10	00000000	00000000	0059	00000000	Down Stopping

```

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

```

```

4 TCP/IP(s) for CS      V2R10 found

```

```

=====

```

Analysis of Tcp/Ip for TCPSVT. Index: 1

```

TCPIP Raw Control Block Analysis
Master Raw Control Block (MRCB)
MRAWCB: 7F75B048
+0000 RMRBCEYE. MRCB      MRCMUTEX. 00000000 00000000 00000000
D7D60501      RSTKDOWN. 00
+0021 RSTKLNKD. 01      RDRVSTAT. 01      RSBCAST.. 00000000
RSDNTRTE. 00000000 RSRVBUF. 0000FFFF

```

```

+0030 RSSNDBUF. 0000FFFF RDIPTOS.. 00      RDIPTTL.. 00
RIPWRQ@.. 7F61D3E8 RIPRDQ@.. 7F61D3A8
+0040 RHASH@... 7F75B08C

```

....

Raw Hash Table Entries

```

ID  First      Last
9   7F5513C8 7F5513C8
15  7F712088 7F712088

```

RCB	ResrcID	ResrcNm	TpiState	DestAddr	ProtocolId
7F5513C8	00000062	OMPROUTE	WLOIDLE	129.11.208.108	89
7F712088	00000008	TCPSVT	WLOIDLE	0.0.0.0	255

```

2 RCB(s) FOUND
2 RCB(s) FORMATTED

```

TCP/IP Analysis

TCPIP Main TCP Control Block (MTCB)

MTCB: 1338E350

```

+0000 M_MAIN_EYE..... TCP MAIN
+0008 M_TCP_LWRITE_Q..... 7F781868
+000C M_TCP_LREAD_Q..... 7F781828
+0014 M_TCP_DRIVER_STATE. 01
+0018 MTCPMTX..... 00000000 00000000 00000000 D7D60601
+0028 MTCPAQMX..... 00000000 00000000 00000000 D7D60604
+0038 MTCB_LIST_LOCK..... 00000000 00000000 00000000 D7D60604
+0048 M_PORT_CEILING..... 00000FFF
+004C M_TPI_SEQ#..... 0001C62B
+0050 M_PORT_ARRAY..... 7F712FC8
+0054 M_LAST_PORT_NUM.... 00000445

```

....

TCB	ResrcID	ResrcNm	TcpState	TpiState	Flag1234	UseCount	IPAddr
Port	LuName	App1Name	UserID				
7F607108	00000002	TCPSVT	Closed	WLOUNBND	00040000	00000001	0.0.0.0
0							
7F60A908	000083D7	FTPUNIX1	Listening	WLOIDLE	00200080	00000001	0.0.0.0
0							
7F608D08	00000013	TCPSVT	Listening	WLOIDLE	00000080	00000001	0.0.0.0
0							
7F617508	0000019B	CICSRU	Listening	WLOIDLE	08200080	00000001	0.0.0.0
0							
7F615108	00000144	INETD5	Listening	WLOIDLE	00200080	00000001	0.0.0.0
0							
...							
7F610108	0000878F	NAMED4	TimeWait	WLOWIORL	80800C00	00000002	198.11.22.103
53							
7F60C508	0000005C	DHCP1	Established	WLOXFER	01800000	00000001	198.11.25.104
6000							
7F609D08	00000049	MISCSRV	Listening	WLOIDLE	00200000	00000001	0.0.0.0
0							
7F608908	00000012	TCPSVT	Listening	WLOIDLE	00000080	00000001	0.0.0.0
0							
7F60E108	00000063	TCPSVT	Established	WLOXFER	80800000	00000001	127.0.0.1
1030							
30 TCB(s)	FOUND						
30 TCB(s)	FORMATTED						

```

User Datagram Protocol Control Block Summary
MUCB: 7F7812A8

```

```

+0000 UMUCBEYE. MUCB      USTKDOWN. 00      USTKLNKD. 01
UAPAR.... 00      UDRVSTAT. 00
+0008 UOPENPRT. 00000000 UFREEPRT. 0408      MCBMUTEX. 00000000
00000000 00000000 D7D60402
+0020 UDPCFG... 00000001 0000FFFF 0000FFFF 00000001 80000000
00000000
+0038 UDPCFG2.. 00000001 0000FFFF 0000FFFF 00000001 80000000
00000000
+0050 UDPMIB... 00001D1F 0000531F 00000000 0000166B
USBCAST.. 00000000 USLPBACK. 00000000
+0068 USDNTRTE. 00000000 USRCVBUF. 0000FFFF USSNDBUF. 0000FFFF
UFGPRC... 00      SERIALV. 0000065F
+007C SERIAL1. 0000065F ULASTADR. 810B2068 ULASTPRT. 0043
ULASTUCB. 7F5FD508 SERIAL2. 0000065F
...

```

UCB	ResrcID	ResrcNm	TpiState	IPAddr	Port
7F5F6108	00000004	TCPSVT	WLOUNBND	0.0.0.0	
7F5FCD08	00000086	OSNMPD	WLOIDLE	127.0.0.1	161
7F5FD508	0000005E	DHCP1	WLOIDLE	129.11.32.1	67
7F5FCF08	00000055	DHCP1	WLOIDLE	198.11.25.104	1027
7F5FD308	0000005B	NAMED	WLOIDLE	129.11.176.87	53
7F5FD108	00000059	DHCP3	WLOIDLE	0.0.0.0	6001
7F5FCB08	0000004B	CBSAMPLE	WLOIDLE	0.0.0.0	30001
7F5FC908	00000048	MISCSRV	WLOIDLE	0.0.0.0	19
7F5FC708	00000047	MISCSRV	WLOIDLE	0.0.0.0	19
....					
7F5F6B08	00000017	MISCSRV	WLOIDLE	0.0.0.0	7
7F5F6908	00000014	PORTMAP	WLOIDLE	0.0.0.0	111
56 UCB(s) FOUND					
56 UCB(s) FORMATTED					

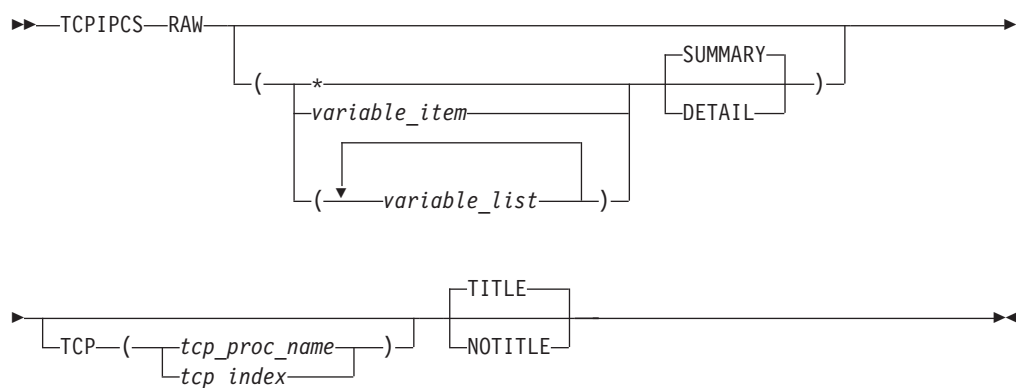
Analysis of Tcp/Ip for TCPSVT completed

TCPIPCS RAW

Invocation of this command displays the Master Raw Control Block (MRCB) and any Raw protocol Control Blocks (RCBs) defined in the MRCB hash table.

Syntax

Following is the syntax of the TCPIPCS RAW subcommand:



Parameters

If no parameters are specified, all raw connections are summarized. The following are the parameters for the TCPIPCS RAW subcommand:

- * An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space, within parentheses:

Variable parameters are:

jobname

Displays only the API control blocks for this job name. The job name may be a TCP/IP application name or a stack name. Must contain from one to eight characters.

RCB_address

Displays only the RCB with this address. An address is specified as one to eight hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

connection_id

Displays the RCB with this connection ID. A connection ID is specified as 1–hexadecimal digit.

In addition to the variable parameters described above, the following keyword parameters may be specified:

SUMMARY

Formats the MRCB contents and lists all the RCBs in one cross-reference table. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL formats the contents of the RCB(s).

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIP CS RAW subcommand:

```
| TCPIP CS RAW
| Dataset: IPCS.R8A0723.RASDUMP
| Title:   EZRPE005
| The address of the TSAB is: 098221F0
| Tseb    SI Procedure Version TsdB    TsdX    Asid TraceOpts Status
| 09822230 1 TCPCS      V1R4    08E85000 08E850C8 001E 9FFF7E7F Active
| 098222B0 2 TCPCS2    V1R4    08937000 089370C8 01F6 9FFF7E7F Active
|      2 defined TCP/IP(s) were found
|      2 active TCP/IP(s) were found
|      2 TCP/IP(s) for CS      V1R4 found
|
| Analysis of Tcp/Ip for TCPCS. Index: 1
| TCPIP Raw Control Block Analysis
| Master Raw Control Block (MRCB)
| MRAWCB: 7F407208
| +0000 RMRCBEYE. MRCB      RSTKLNKD. 01      RDRVSTAT. 01
| +000C R6STKLNKD.          01
| +000D R6DRVSTAT.          01
| +0010 MRCMUTEX. 00000000 00000000 00000000 D7D60501
```

```

+002C  RSBCAST.. 00000000  RSDNTRTE. 00000000  RSRC
      RSSNDBUF. 0000FFFF  RDIPTOS.. 00      RDIPTTL.. 00
      RIPWRQ@.. 7F621DA8  RIPRDQ@.. 7F621D68  RHAS
+0040  RIP6WRQ@. 7F686468  RIP6RDQ@. 7F686428  R6HASH@.. 7F407374
+004C  R6DFFLTR. 7F781FFF  FFFFFFFF  FFFFFFFF  FFFFFFFF  003FFFFFF
      FFFFFFFF  FFFFFFFF  FFFFFFFF

IPv4 Raw Hash Table Entries
ID  First      Last
0   7F52C390  7F52C390
15  7F52C110  7F52C110

IPv6 Raw Hash Table Entries
ID  First      Last
0   7F2073BC  7F2073BC

IPv4 RAW Connections
RCB      ResrcID  ResrcNm  TpiState  ProtocolId  DestAddr
7F52C388 00000006  TCPCS    WLOIDLE   0            0.0.0.0
7F52C108 00000008  TCPCS    WLOIDLE   255          0.0.0.0

IPv6 RAW Connections
RCB      ResrcID  ResrcNm  TpiState  ProtocolId  DestAddr
7F207208 0000000E  TCPCS    WLOIDLE   0            ::0

3 RCB(s) FOUND
3 RCB(s) FORMATTED

Analysis of Tcp/Ip for TCPCS completed

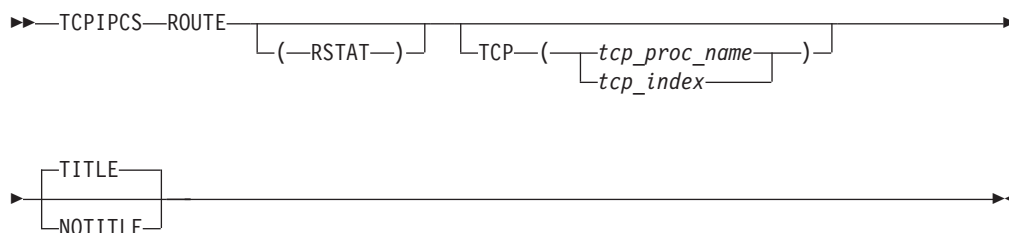
```

TCPIPCS ROUTE

Invocation of this subcommand displays the routing control blocks. Each routing table entry is formatted to display the address device name, type, protocol, destination IP address, gateway IP address, and the physical interface control block address.

Syntax

Following is the syntax of the TCPIPCS ROUTE subcommand:



Parameters

Following are the parameters for the TCPIPCS ROUTE subcommand:

ALL

Display structure of all route table information. ALL is the default.

IPV4

All IPv4 search tree and update tree routes.

IPV6

All IPv6 search tree and update tree routes.

IQDIO

All iQDIO search tree and update tree routes.

RSTAT

All defined replaceable static routes are displayed without regard to whether or not they are currently being used in the active routing table.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set (ALL, IPV4, IPV6, IQDIO, RSTAT), all of them will be used.

Sample output

The following is sample output of the TCPIPCS ROUTE subcommand:

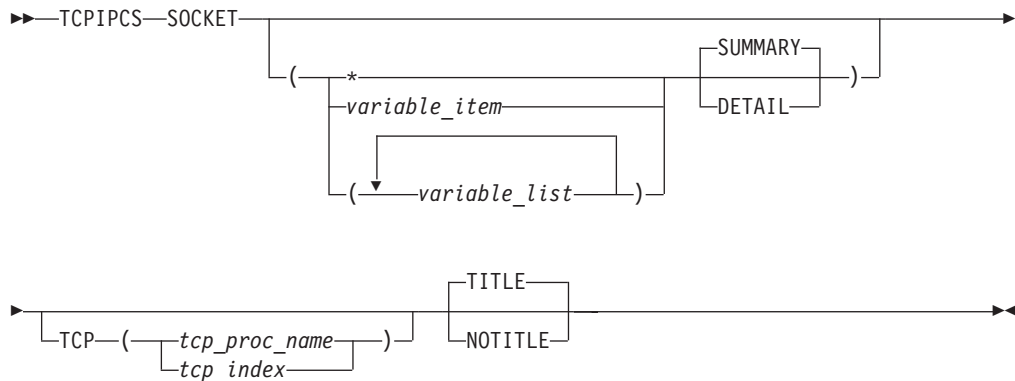
```
TCPIPCS ROUTE ( IPV4 IPV6 ) TCP ( 1 )
Dataset: IPCS.MV25332.DUMPC
Title: SLIP DUMP ID=TK02
The address of the TSAB is: 1EF11570
Tseb SI Procedure Version Tsdb TsdX Asid TraceOpts Status
1EF115B0 1 TCPSVT V1R4 1DE47000 1DE470C8 0057 C5BF755F Active
1EF11630 2 TCPSVT3 V1R4 1DD4D000 1DD4D0C8 0074 C5BF755F Active
2 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
2 TCP/IP(s) for CS V1R4 found
=====
Analysis of Tcp/Ip for TCPSVT. Index: 1
TCPIP Route Analysis
IPv4 Routes in Search Table
Rte@ DeviceName Type Protocol Pif@ IP Addresses
7EED5708 LCGE2SP32 Host OSPF 7F539088 Destination: 216.51.55.202
Gateway : 198.11.32.110
7EEB7888 LMGE2RU33 Host OSPF 7F538608 Destination: 216.51.55.202
Gateway : 198.11.33.104
7EF0DBE8 LCGE2SP32 Host OSPF 7F539088 Destination: 216.51.55.155
Gateway : 198.11.32.110
IPv6 Routes in Search Table
Rte@ DeviceName Type Protocol Pif@ IP Addresses
7EE718E8 LV60GETH2 Direct ICMP 7F53C888 Destination: FEC0:176:11:48::0
Gateway : ::0
7EE46068 LV60GETH2 Default ICMP 7F53C888 Destination: ::0
Gateway : FE80::230:71FF:FED3:5160
IPv4 Routes in Update Table
Rte@ DeviceName Type Protocol Pif@ IP Addresses
7EED5708 LCGE2SP32 Host OSPF 7F539088 Destination: 216.51.55.202
Gateway : 198.11.32.110
7EEB7888 LMGE2RU33 Host OSPF 7F538608 Destination: 216.51.55.202
Gateway : 198.11.33.104
7EF0DBE8 LCGE2SP32 Host OSPF 7F539088 Destination: 216.51.55.155
Gateway : 198.11.32.110
7EEB7A08 LMGE2RU33 Host OSPF 7F538608 Destination: 216.51.55.155
Gateway : 198.11.33.104
7EED5A08 LCGE2SP32 Host OSPF 7F539088 Destination: 216.22.1.202
Gateway : 198.11.32.110
7EEB7B88 LMGE2RU33 Host OSPF 7F538608 Destination: 216.22.1.202
Gateway : 198.11.33.104
IPv6 Routes in Update Table
Rte@ DeviceName Type Protocol Pif@ IP Addresses
7EE718E8 LV60GETH2 Direct ICMP 7F53C888 Destination: FEC0:176:11:48::0
Gateway : ::0
7EE46068 LV60GETH2 Default ICMP 7F53C888 Destination: ::0
Gateway : FE80::230:71FF:FED3:5160
Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS SOCKET

Invocation of this command displays information from AF_UNIX socket control blocks.

Syntax

Following is the syntax of the TCPIP CS SOCKET subcommand:



Parameters

If no parameters are specified, all sockets are summarized.

- * An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters may be repeated, each separated by a blank space, within parentheses:

Variable parameters are:

SCB_address

Displays only the socket control block (SCB) with this address. An address is specified as one to eight hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

connection_id

Displays the SCB with this connection ID. A connection ID is specified as one to eight hexadecimal digits.

In addition to the variable parameters described above, the following keyword parameters can be specified:

SUMMARY

Summarizes the sockets. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL formats the contents of the SCBs.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIP CS SOCKET subcommand:

TCPIPCS SOCKET
Dataset: IPCS.MV21381.DUMPA
Title: SLIP DUMP ID=TC

The address of the TSAB is: 13391BC0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
13391C00	1	TCPSVT	V2R10	1323B000	1323B0C8	07DE	04041405	Active
13391C80	2	TCPSVT2	V2R10	00000000	00000000	07E8	00000000	Down Stopping
13391D00	3	TCPSVT1	V2R10	12FC3000	12FC30C8	0080	94FF755F	Active
13391D80	4	TCPSVT3	V2R10	00000000	00000000	0059	00000000	Down Stopping

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

4 TCP/IP(s) for CS V2R10 found

=====

Analysis of Tcp/Ip for TCPSVT. Index: 1

TCPIP Socket Analysis

SCB	CID	Protocol	SockOpts	ScbFlags	ResrcNm
12D40108	00000008	RAW	00000000	00280000	TCPSVT
12D40208	0000000B	UDP	00000000	00280000	TCPSVT
12D40308	0000000C	TCP	00020000	C0280000	TCPSVT
12D40408	0000000E	UDP	00000000	00280000	TCPSVT
12D40508	0000000F	TCP	00000000	B0280000	TCPSVT
12D40608	00000010	TCP	00020000	90280000	TCPSVT
12D40708	00000067	TCP	08000000	90280000	OMPROUTE
12D40808	00000012	TCP	00400000	C0000000	TCPSVT
12D40908	00000013	TCP	00400000	C0000000	TCPSVT
12D40A08	00000014	UDP	00000000	80280000	PORTMAP
12D40B08	00000015	TCP	00000000	C0280000	PORTMAP
...					
12D44C08	00000058	TCP	00000000	C0000000	DHCP3
12D44D08	00000059	UDP	00000000	80280000	DHCP3
12D44E08	0000005A	TCP	00400000	C0280000	NAMED
12D44F08	0000005B	UDP	00400000	80280000	NAMED

79 Socket control blocks were found
79 Socket control blocks were formatted

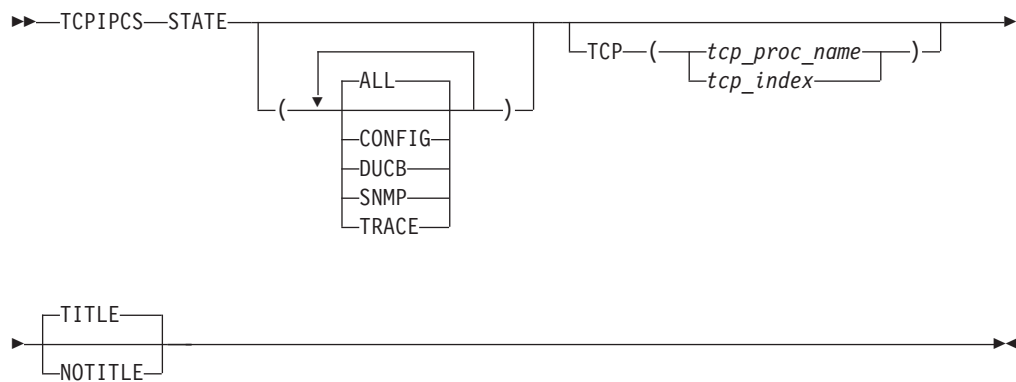
Analysis of Tcp/Ip for TCPSVT completed

TCPIPCS STATE

Invocation of this subcommand provides an overall view of TCP/IP. Major control block addresses, subtasks, storage usage, dispatchable units, trace, and configuration are displayed.

Syntax

Following is the syntax of the TCPIPCS STATE subcommand:



Parameters

The following keyword parameters can be specified:

ALL

Display all state information. ALL is the default.

CONFIG

Only display configuration state information.

DUCEB

Only display DUCB state information.

SNMP

Only display SNMP and CONFIG information. (SNMP information makes sense only in the context of the configuration, so the configuration information is also be displayed.)

TRACE

Only display trace state information.

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {ALL, CONFIG, DUCB, SNMP, TRACE}, all of them will be used.

Sample output

The following is sample output of the TCPIP CS STATE subcommand:

```

TCPIP CS STATE
Dataset: IPCS.R450697.V6TCBD1
Title:  TCPCS2 CLIENT SIDE

```

The address of the TSAB is: 09DBE1A0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
09DBE1E0	1	TCPCS	V1R4	096C4000	096C40C8	0033	10841004	Active
09DBE260	2	TCPCS2	V1R4	096C9000	096C90C8	0034	10841004	Active

```

2 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

```

```

2 TCP/IP(s) for CS      V1R4  found

```

```

=====

```

Analysis of Tcp/Ip for TCPCS2. Index: 2

TCP/IP State

TCP/IP Status:

```

Procedure:   TCPCS2
Version:     V1R4
Status:      Active
Asid:        0034
Started:     2001/11/07 17:34:16
Ended:       2001/11/07 17:41:23
Active:      00:07:06.980607 hours

```

Major Control Blocks

```

TSEB:        9DBE260      TSDB:        96C9000
TSDX:        96C90C8      TCA:        97E9610
ITCVT:       96C93C8      ITSTOR:     96C95E0
DUAF:        92F0010      MRCB:       7F40E208
MTCB:        96CE910      MUCB:       7F512248
IPMAIN:      95A83D0      Streams_root: 7F6BFD08
TosMains:    96CE770      MIB2:       97E9010
CdCb:        97E9390      User:       9EDD060
Conf:        96CC110      Stks:       9D098B0
IPMAIN6:     96CE470

```

TCP/IP Subtasks

Task	Tcb	FirstRB	EotECB	StopEcb	CmpCode	RsnCode	RTWA
EZBTCPIP	007EC920	007ECF98	807FD178		00000000	00000000	00000000
EPWPITSK	007EC690	007EC608	00000000		00000000	00000000	00000000
EZBITTUB	007EC400	007E2AB8	00000000	807EC0E8	00000000	00000000	00000000
EZBIPSUB	007EC170	007EC060	00000000	807EC060	00000000	00000000	00000000
EZBIEOER	007E3D90	007E3D08	00000000	807E3D08	00000000	00000000	00000000
EZACDMSM	007E3B70	007E3A88	00000000	807E3A88	00000000	00000000	00000000
EZACFMMN	007E38F0	007E3838	807ECF98	807E3838	00000000	00000000	00000000
EZBTZMST	007E34F0	007E33D0	007ECF98		00000000	00000000	00000000
EZBTSSL	007E3140	007E30B8	807E33D0		00000000	00000000	00000000
EZBTMCTL	007E2D90	007E32F8	807E33D0		00000000	00000000	00000000
EZACFALG	007E2BF8	007E2B60	807ECF98		00000000	00000000	00000000
EZASASUB	007E28D0	007E2848	807ECF98	807E2848	00000000	00000000	00000000

Storage Cache Information

```

Total CSA Allocated:      4,959,664
Tcp/ip CSA Limit:         2,147M
Total CSA Elements:       52
Cache Delay:              180 seconds
Scan Delay:               75 seconds
Total cache allocated:    22,040
Total cache elements:     3
Total freed elements:     0
Last cache scan time:     2001/11/07 21:39:45

```

CSM Status

```

ECSA Storage:            OK
Data Space Storage:      OK
Fixed Storage:           OK
Alet: 01FF0009           Dspname: CSM31001

```

Dispatchable Unit Status

```

DUCB Initializations:    7,091
DUCB Expansions:         63
Percent DUCB expansions: 0 %
Last DUCB scan time:     2001/11/07 21:40:48

```

```

1 DUAT control block(s) were found in the DUAF at 092F0010
82 Dispatchable units were found.
No DUs indicate abend.

```

```

CTrace Status:
Member Name : CTIEZBN0
Buffer Size : 20,972K
Options      : Socket PFS TCP Internet VtamData
Asid List    : ()
JobNameList  : ()
PortList     : ()
IpAddrList   : ()
Xwriter      : Disconnected
Dwriter      : Disconnected
Trace Count  : 313,343
Lost Count   : 0
Lost Time    : 1900/01/01 00:00:00
Wrap Count   : 10
Wrap Time    : 2001/11/07 21:40:51

```

```

=====
Device Interface: 7F6AA408
Device: LOOPBACK      Devtype: LOOPBACK      State: Active
Address: **** ****

```

```

Physical Interface: 7F679468
Name: LOOPBACK      Protocol: LOOPBACK      State: Active
NetNum: 0  QueueSize: 0  Bytein: 3,861  Byteout: 3,861
Index: 2
Bsd Routing Parameters:
MtuSize: 0          Metric: 0
SubnetMask: 0.0.0.0  DestAddr: 0.0.0.0
SNMP Input Counters:
  Octets:          3,861  Unicast:          61
  NonUnicast:      0      Discarded:         0
  Error:           0      Unkn Type:         0
  Broadcast:       0      Multicast:         0
SNMP Output Counters:
  Octets:          3,861  Unicast:          61
  NonUnicast:      0      Discarded:         0
  Error:           0      Queue Len:         0
  Broadcast:       0      Multicast:         0

```

```

IPv4 Search Patricia tree  Address: 7F679F68
Search Ptree Reader Count: 0

```

```

Route: 7F6AA288
Name: LOOPBACK      Type: Direct      State: Active
Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
Protocol : Configuration  Gate: 0.0.0.0
Mtu Size: 65535      Ref Cnt: 2      Tos: 0
Metric1: 0           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:25

```

```

IPv6 Search Patricia tree  Address: 7F3FDF08
Search Ptree Reader Count: 0

```

```

IPv4 Update Patricia tree  Address: 7F679F08
No update to Route Ptree is pending.

```

```

Route: 7F6AA288
Name: LOOPBACK      Type: Direct      State: Active
Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
Protocol : Configuration  Gate: 0.0.0.0
Mtu Size: 65535      Ref Cnt: 2      Tos: 0
Metric1: 0           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:25

```

```

IPv6 Update Patricia tree  Address: 7F3FDF08
No update to Route Ptree is pending.

```



```

Logical Interface: 7F6792E8
  Name: LOOPBACK          Protocol: LOOPBACK      State: Active
  Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
  Mtu Size: 65535
  Packet Trace Parameters: 7F1AFA48
    Protocol: 0    TrRecCnt: 48    PckLength: 65535
    SrcPort: 0     DestPort: 0
    SubnetMask: 255.255.255.255  DestAddr: 0.0.0.0
Physical Interface: 7F503488
  Name: LOOPBACK6         Protocol: LOOPBACK6    State: Active
  NetNum: 0    QueSize: 0    Bytein: 0        Byteout: 0
  Index: 3
  Bsd Routing Parameters:
    MtuSize: 0            Metric: 0
    SubnetMask: 0.0.0.0    DestAddr: 0.0.0.0
  SNMP Input Counters:
    Octets:                0    Unicast:          0
    NonUnicast:            0    Discarded:        0
    Error:                 0    Unkn Type:        0
    Broadcast:             0    Multicast:        0
  SNMP Output Counters:
    Octets:                0    Unicast:          0
    NonUnicast:            0    Discarded:        0
    Error:                 0    Queue Len:        0
    Broadcast:             0    Multicast:        0

IPv4 Search Patricia tree  Address: 7F679F68
Search Ptree Reader Count: 0
IPv6 Search Patricia tree  Address: 7F3FDF08
Search Ptree Reader Count: 0

```

```

Route: 7F3FD9C8
  Name: LOOPBACK6         Type: Direct          State: Active
  Subnet Prefix: 128      Addr: ::1
  Protocol : Configuration Gate: ::0
  Mtu Size: 0            Ref Cnt: 0          Tos: 28
  Metric1: 0             Metric2: -1
  Metric3: -1            Metric4: -1
  Metric5: -1            Age: 2001/11/07 21:34:28

```

```

IPv4 Update Patricia tree  Address: 7F679F08
No update to Route Ptree is pending.
IPv6 Update Patricia tree  Address: 7F3FDF08
No update to Route Ptree is pending.

```

```

Route: 7F3FD9C8
  Name: LOOPBACK6         Type: Direct          State: Active
  Subnet Prefix: 128      Addr: ::1
  Protocol : Configuration Gate: ::0
  Mtu Size: 0            Ref Cnt: 0          Tos: 28
  Metric1: 0             Metric2: -1
  Metric3: -1            Metric4: -1
  Metric5: -1            Age: 2001/11/07 21:34:28

```

```

Logical Interface: 7F3FDCE8
  Name: LOOPBACK6          Protocol: LOOPBACK6    State: Active
  Subnet Prefix: 128        Addr: ::1
  Mtu Size: 65535
  Packet Trace Parameters: 7F1AFB88
    Protocol: 0    TrRecCnt: 0    PckLength: 65535
    SrcPort: 0     DestPort: 0
    SubnetMask: 255.255.255.255  DestAddr: ::0
=====

```

```

Device Interface: 7F1ED408
Device: OSAQDI03          Devtype: MPCIPA        State: Active

```

```

Address: **** ****
SAP:
  UserID: 10010000    TransId: 0001012F    ProviderId: 00010131
  Data@: 8928E0CC    ReqSignal@: 85A2F810    RspSignal@: 85A2F810
  State: Active      Retry: 0    Restart: 0    Xstatus: 0
Connection 2:
  UserID: 00000000    ProviderId: 90010001
  Data@: 00000000    ReqSignal@: 00000000    RspSignal@: 00000000
  State: Reset      linknum: 00    flags 00

Physical Interface: 7F503B88
  Name: OSAQDIOL      Protocol: IPAQENET      State: Active
  NetNum: 0    QueSize: 0    Bytein: 0    Byteout: 0
  Index: 5
  Bsd Routing Parameters:
    MtuSize: 0      Metric: 0
    SubnetMask: 255.0.0.0    DestAddr: 0.0.0.0
  SNMP Input Counters:
    Octets: 0    Unicast: 0
    NonUnicast: 0    Discarded: 0
    Error: 0    Unkn Type: 0
    Broadcast: 0    Multicast: 0
  SNMP Output Counters:
    Octets: 0    Unicast: 0
    NonUnicast: 0    Discarded: 0
    Error: 0    Queue Len: 0
    Broadcast: 0    Multicast: 0

IPv4 Search Patricia tree    Address: 7F679F68
  Search Ptree Reader Count: 0

Route: 7F3FD188
  Name: OSAQDIOL      Type: Direct      State: Active
  Subnet Mask: 255.255.255.255    Addr: 9.67.115.83
  Protocol : Configuration    Gate: 0.0.0.0
  Mtu Size: 8992    Ref Cnt: 0    Tos: 0
  Metric1: 0      Metric2: -1
  Metric3: -1      Metric4: -1
  Metric5: -1      Age: 2001/11/07 21:34:31

Route: 7F3081E8
  Name: OSAQDIOL      Type: Direct      State: Active
  Subnet Mask: 255.255.255.255    Addr: 9.67.115.79
  Protocol : Configuration    Gate: 0.0.0.0
  Mtu Size: 1500    Ref Cnt: 0    Tos: 0
  Metric1: 0      Metric2: -1
  Metric3: -1      Metric4: -1
  Metric5: -1      Age: 2001/11/07 21:34:31

Route: 7F1EB128
  Name: OSAQDIOL      Type: Direct      State: Active
  Subnet Mask: 255.255.255.255    Addr: 9.67.115.82
  Protocol : Configuration    Gate: 0.0.0.0
  Mtu Size: 8992    Ref Cnt: 0    Tos: 0
  Metric1: 0      Metric2: -1
  Metric3: -1      Metric4: -1
  Metric5: -1      Age: 2001/11/07 21:34:31

Route: 7F628348
  Name: OSAQDIOL      Type: Direct      State: Active
  Subnet Mask: 255.255.255.192    Addr: 9.67.115.0
  Protocol : Configuration    Gate: 0.0.0.0
  Mtu Size: 8992    Ref Cnt: 0    Tos: 0
  Metric1: 0      Metric2: -1
  Metric3: -1      Metric4: -1
  Metric5: -1      Age: 2001/11/07 21:34:31

```

```

Route: 7F6AA108
Name: OSAQDIOL          Type: Default          State: Active
Subnet Mask: 0.0.0.0      Addr: 0.0.0.0
Protocol : Configuration  Gate: 9.67.115.1
Mtu Size: 8992           Ref Cnt: 0          Tos: 0
Metric1: 1               Metric2: -1
Metric3: -1              Metric4: -1
Metric5: -1              Age: 2001/11/07 21:34:31
IPv6 Search Patricia tree Address: 7F3FDF08
Search Ptree Reader Count: 0

IPv4 Update Patricia tree Address: 7F679F08
No update to Route Ptree is pending.

Route: 7F3FD188
Name: OSAQDIOL          Type: Direct          State: Active
Subnet Mask: 255.255.255.255 Addr: 9.67.115.83
Protocol : Configuration  Gate: 0.0.0.0
Mtu Size: 8992           Ref Cnt: 0          Tos: 0
Metric1: 0               Metric2: -1
Metric3: -1              Metric4: -1
Metric5: -1              Age: 2001/11/07 21:34:31

Route: 7F3081E8
Name: OSAQDIOL          Type: Direct          State: Active
Subnet Mask: 255.255.255.255 Addr: 9.67.115.79
Protocol : Configuration  Gate: 0.0.0.0
Mtu Size: 1500           Ref Cnt: 0          Tos: 0
Metric1: 0               Metric2: -1
Metric3: -1              Metric4: -1
Metric5: -1              Age: 2001/11/07 21:34:31

Route: 7F1EB128
Name: OSAQDIOL          Type: Direct          State: Active
Subnet Mask: 255.255.255.255 Addr: 9.67.115.82
Protocol : Configuration  Gate: 0.0.0.0
Mtu Size: 8992           Ref Cnt: 0          Tos: 0
Metric1: 0               Metric2: -1
Metric3: -1              Metric4: -1
Metric5: -1              Age: 2001/11/07 21:34:31

Route: 7F628348
Name: OSAQDIOL          Type: Direct          State: Active
Subnet Mask: 255.255.255.192 Addr: 9.67.115.0
Protocol : Configuration  Gate: 0.0.0.0
Mtu Size: 8992           Ref Cnt: 0          Tos: 0
Metric1: 0               Metric2: -1
Metric3: -1              Metric4: -1
Metric5: -1              Age: 2001/11/07 21:34:31

Route: 7F6AA108
Name: OSAQDIOL          Type: Default          State: Active
Subnet Mask: 0.0.0.0      Addr: 0.0.0.0
Protocol : Configuration  Gate: 9.67.115.1
Mtu Size: 8992           Ref Cnt: 0          Tos: 0
Metric1: 1               Metric2: -1
Metric3: -1              Metric4: -1
Metric5: -1              Age: 2001/11/07 21:34:31
IPv6 Update Patricia tree Address: 7F3FDF08
No update to Route Ptree is pending.

Address Translate Entry: 7F1AF5E8
addr: 9.67.115.79      flags: C0 ttl: 0
retries: 0
Address Translate Entry: 7F1AF548
addr: 9.67.115.82      flags: C0 ttl: 0
retries: 0

```

Address Translate Entry: 7F1AF4A8
addr: 9.67.115.1 flags: C0 ttl: 0
retries: 0

Logical Interface: 7F1ED008

Name: OSAQDIOL Protocol: IPAQENET State: Active
Subnet Mask: 255.255.255.255 Addr: 9.67.115.82
Mtu Size: 1492
Packet Trace Parameters: 7F1AF4A8
Protocol: 0 TrRecCnt: 0 PckLength: 65535
SrcPort: 0 DestPort: 0
SubnetMask: 255.255.255.255 DestAddr: 0.0.0.0

Physical Interface: 7F503F08

Name: OSAQDI26 Protocol: IPAQENET6 State: Active
NetNum: 0 QueSize: 0 Bytein: 54,570K Byteout: 285,212
Index: 6

Bsd Routing Parameters:

MtuSize: 0 Metric: 0
SubnetMask: 0.0.0.0 DestAddr: 0.0.0.0

SNMP Input Counters:

Octets:	54,570K	Unicast:	29,240
NonUnicast:	0	Discarded:	0
Error:	0	Unkn Type:	0
Broadcast:	0	Multicast:	7

SNMP Output Counters:

Octets:	285,212	Unicast:	2,401
NonUnicast:	0	Discarded:	0
Error:	0	Queue Len:	0
Broadcast:	0	Multicast:	12

IPv4 Search Patricia tree Address: 7F679F68

Search Ptree Reader Count: 0

IPv6 Search Patricia tree Address: 7F3FDF08

Search Ptree Reader Count: 0

Route: 7F1EBE88

Name: OSAQDI26 Type: Host State: Active
Subnet Prefix: 128 Addr: FEC9:C2D4:1::9:67:115:83
Protocol : Configuration Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
Mtu Size: 0 Ref Cnt: 0 Tos: 28
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2001/11/07 21:34:31

Route: 7F1EC3E8

Name: OSAQDI26 Type: Direct State: Active
Subnet Prefix: 128 Addr: FEC9:C2D4:1::9:67:115:79
Protocol : Configuration Gate: ::0
Mtu Size: 5 Ref Cnt: 0 Tos: 0
Metric1: 0 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2001/11/07 21:34:31

Route: 7F1C7C48

Name: OSAQDI26 Type: Direct State: Active
Subnet Prefix: 128 Addr: FEC9:C2D4:1::9:67:115:82
Protocol : Configuration Gate: ::0
Mtu Size: 0 Ref Cnt: 0 Tos: 28
Metric1: 0 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2001/11/07 21:34:36

Route: 7F1ECE88

Name: OSAQDI26 Type: Direct State: Active
Subnet Prefix: 128 Addr: FEC9:C2D4:1::206:2AFF:FE66:C81C
Protocol : Configuration Gate: ::0
Mtu Size: 0 Ref Cnt: 0 Tos: 0

```

Metric1: 0           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:31

Route: 7F1C7E88
Name: OSAQDI26       Type: Direct           State: Active
Subnet Prefix: 128   Addr: FE80::2:559A:3F5F:1
Protocol : Configuration Gate: ::0
Mtu Size: 2          Ref Cnt: 0             Tos: 28
Metric1: 0           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:32

Route: 7F1ECAE8
Name: OSAQDI26       Type: Direct           State: Active
Subnet Prefix: 128   Addr: 50C9:C2D4:1::206:2AFF:FE66:C81C
Protocol : Configuration Gate: ::0
Mtu Size: 0          Ref Cnt: 0             Tos: 0
Metric1: 0           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:31

Route: 7F1EB588
Name: OSAQDI26       Type: Prefix           State: Active
Subnet Prefix: 112   Addr: 50C9:C2D4:1::0
Protocol : Configuration Gate: 50C9:C2D4:1::206:2AFF:FE66:C81C
Mtu Size: 0          Ref Cnt: 0             Tos: 28
Metric1: 1           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:31

Route: 7F1C7768
Name: OSAQDI26       Type: Direct           State: Active
Subnet Prefix: 64    Addr: 50C9:C2D4:1::0
Protocol : ICMP       Gate: ::0
Mtu Size: 0          Ref Cnt: 0             Tos: 28
Metric1: 0           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:57

Route: 7F1C74C8
Name: OSAQDI26       Type: Direct           State: Active
Subnet Prefix: 64    Addr: 50C9:FFFF:1::0
Protocol : ICMP       Gate: ::0
Mtu Size: 0          Ref Cnt: 0             Tos: 28
Metric1: 0           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:57

Route: 7F1EBA28
Name: OSAQDI26       Type: Prefix           State: Active
Subnet Prefix: 112   Addr: FEC9:C2D4:1::0
Protocol : Configuration Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
Mtu Size: 0          Ref Cnt: 0             Tos: 28
Metric1: 1           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:31

Route: 7F1C7A08
Name: OSAQDI26       Type: Direct           State: Active
Subnet Prefix: 64    Addr: FEC9:C2D4:1::0
Protocol : ICMP       Gate: ::0
Mtu Size: 0          Ref Cnt: 0             Tos: 28
Metric1: 0           Metric2: -1
Metric3: -1          Metric4: -1
Metric5: -1          Age: 2001/11/07 21:34:57

```

```

Route: 7F1EC748
Name: OSAQDI26          Type: Default          State: Active
Subnet Prefix: 0        Addr: ::0
Protocol : Configuration Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
Mtu Size: 0            Ref Cnt: 0            Tos: 28
Metric1: 1             Metric2: -1
Metric3: -1            Metric4: -1
Metric5: -1            Age: 2001/11/07 21:34:31

IPv4 Update Patricia tree Address: 7F679F08
No update to Route Ptree is pending.
IPv6 Update Patricia tree Address: 7F3FDF08
No update to Route Ptree is pending.

Route: 7F1EBE88
Name: OSAQDI26          Type: Host          State: Active
Subnet Prefix: 128      Addr: FEC9:C2D4:1::9:67:115:83
Protocol : Configuration Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
Mtu Size: 0            Ref Cnt: 0            Tos: 28
Metric1: 1             Metric2: -1
Metric3: -1            Metric4: -1
Metric5: -1            Age: 2001/11/07 21:34:31

Route: 7F1EC3E8
Name: OSAQDI26          Type: Direct        State: Active
Subnet Prefix: 128      Addr: FEC9:C2D4:1::9:67:115:79
Protocol : Configuration Gate: ::0
Mtu Size: 5            Ref Cnt: 0            Tos: 0
Metric1: 0             Metric2: -1
Metric3: -1            Metric4: -1
Metric5: -1            Age: 2001/11/07 21:34:31

Route: 7F1C7C48
Name: OSAQDI26          Type: Direct        State: Active
Subnet Prefix: 128      Addr: FEC9:C2D4:1::9:67:115:82
Protocol : Configuration Gate: ::0
Mtu Size: 0            Ref Cnt: 0            Tos: 28
Metric1: 0             Metric2: -1
Metric3: -1            Metric4: -1
Metric5: -1            Age: 2001/11/07 21:34:36

Route: 7F1ECE88
Name: OSAQDI26          Type: Direct        State: Active
Subnet Prefix: 128      Addr: FEC9:C2D4:1::206:2AFF:FE66:C81C
Protocol : Configuration Gate: ::0
Mtu Size: 0            Ref Cnt: 0            Tos: 0
Metric1: 0             Metric2: -1
Metric3: -1            Metric4: -1
Metric5: -1            Age: 2001/11/07 21:34:31

Route: 7F1C7E88
Name: OSAQDI26          Type: Direct        State: Active
Subnet Prefix: 128      Addr: FE80::2:559A:3F5F:1
Protocol : Configuration Gate: ::0
Mtu Size: 2            Ref Cnt: 0            Tos: 28
Metric1: 0             Metric2: -1
Metric3: -1            Metric4: -1
Metric5: -1            Age: 2001/11/07 21:34:32

Route: 7F1ECAE8
Name: OSAQDI26          Type: Direct        State: Active
Subnet Prefix: 128      Addr: 50C9:C2D4:1::206:2AFF:FE66:C81C
Protocol : Configuration Gate: ::0
Mtu Size: 0            Ref Cnt: 0            Tos: 0
Metric1: 0             Metric2: -1
Metric3: -1            Metric4: -1
Metric5: -1            Age: 2001/11/07 21:34:31

```

```

Route: 7F1EB588
Name: OSAQDI26          Type: Prefix          State: Active
Subnet Prefix: 112      Addr: 50C9:C2D4:1::0
Protocol : Configuration Gate: 50C9:C2D4:1::206:2AFF:FE66:C81C
Mtu Size: 0             Ref Cnt: 0             Tos: 28
Metric1: 1              Metric2: -1
Metric3: -1             Metric4: -1
Metric5: -1             Age: 2001/11/07 21:34:31

Route: 7F1C7768
Name: OSAQDI26          Type: Direct          State: Active
Subnet Prefix: 64      Addr: 50C9:C2D4:1::0
Protocol : ICMP         Gate: ::0
Mtu Size: 0            Ref Cnt: 0             Tos: 28
Metric1: 0              Metric2: -1
Metric3: -1             Metric4: -1
Metric5: -1             Age: 2001/11/07 21:34:57

Route: 7F1C74C8
Name: OSAQDI26          Type: Direct          State: Active
Subnet Prefix: 64      Addr: 50C9:FFFF:1::0
Protocol : ICMP         Gate: ::0
Mtu Size: 0            Ref Cnt: 0             Tos: 28
Metric1: 0              Metric2: -1
Metric3: -1             Metric4: -1
Metric5: -1             Age: 2001/11/07 21:34:57

Route: 7F1EBA28
Name: OSAQDI26          Type: Prefix          State: Active
Subnet Prefix: 112      Addr: FEC9:C2D4:1::0
Protocol : Configuration Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
Mtu Size: 0             Ref Cnt: 0             Tos: 28
Metric1: 1              Metric2: -1
Metric3: -1             Metric4: -1
Metric5: -1             Age: 2001/11/07 21:34:31

Route: 7F1C7A08
Name: OSAQDI26          Type: Direct          State: Active
Subnet Prefix: 64      Addr: FEC9:C2D4:1::0
Protocol : ICMP         Gate: ::0
Mtu Size: 0            Ref Cnt: 0             Tos: 28
Metric1: 0              Metric2: -1
Metric3: -1             Metric4: -1
Metric5: -1             Age: 2001/11/07 21:34:57

Route: 7F1EC748
Name: OSAQDI26          Type: Default          State: Active
Subnet Prefix: 0        Addr: ::0
Protocol : Configuration Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
Mtu Size: 0             Ref Cnt: 0             Tos: 28
Metric1: 1              Metric2: -1
Metric3: -1             Metric4: -1
Metric5: -1             Age: 2001/11/07 21:34:31

Logical Interface: 7F6BF028
Name: OSAQDI26          Protocol: IPAQENET6    State: Active
Subnet Prefix: 128      Addr: FE80::2:559A:3F5F:1
Mtu Size: 8992
Packet Trace Parameters: 7F1AFC28
Protocol: 0             TrRecCnt: 31646      PckLength: 65535
SrcPort: 0              DestPort: 0
SubnetMask: 255.255.255.255 DestAddr: ::0

Logical Interface: 7F3083C8
Name: OSAQDI26          Protocol: IPAQENET6    State: Active
Subnet Prefix: 0        Addr: FEC9:C2D4:1::9:67:115:82

```

Mtu Size: 1492

Analysis of Tcp/Ip for TCPCS2 completed

TCIPCS STORAGE

Invocation of the command displays the TCP/IP storage summary referenced in common cached storage.

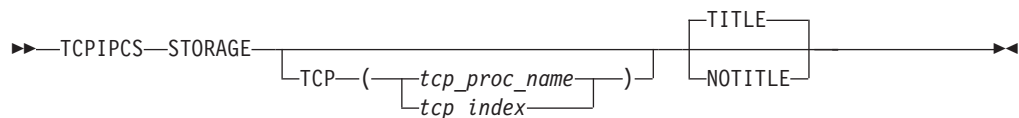
Under the heading Storage Summary, a "c" in column "c" indicates the address is on the cache queue. A "p" in column "p" indicates that the control block is part of a pool.

Cache storage has 12 bytes from offset four overlaid with a chain pointer and time stamp. This may show incorrect data for cached control blocks.

Note: The TCIPCS STORAGE command only reports storage found in caches in common storage. Use the TCIPCS MAP command to report both common and TCP/IP private storage usage.

Syntax

Following is the syntax of the TCIPCS STORAGE subcommand:



Parameters

Following are the parameters for the TCIPCS STORAGE subcommand:

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Note: When a BLS18100I message indicating an access failure appears in the report, any counts or analysis dependent on this information cannot be included in the TCIPCS STORAGE output. Also, an access failure can occur as a result of insufficient user region size. If a BLS18100I message is received for data that is included in the dump, increase the user region size and attempt the TCIPCS STORAGE subcommand again.

Sample output

The following is sample output of the TCIPCS STORAGE subcommand:

```
TCIPCS STORAGE
Dataset: IPCS.A594094.DUMPM
Title:  TCPSVT  V2R10: Job(TCPSVT ) EZBITST0(HTCP50A 99.281)+
       00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051
...
TCPIP Storage Analysis

Storage Statistics
cache_delay           0 seconds before cache is freed
com_totstor          177,578,656 total storage for CSA elements
com_totelem           21,469 total number of CSA elements
scan_delay            120 seconds between full scans
stor_cache            48,416 storage in cache after scan
num_cache             11 elements in cache after scan
num_freed              2 elements freed during last scan
scan_time             1999/10/24 04:06:12 time of last scan
```



```

dsa_init          10,375,262 # of DUCB initializations
dsa_exp           2,180,028 # of DUCB expansions
The control block at 008AC010 (Prev: 00000000) has already been added
...
The control block at 12A26410 (Prev: 137CB0A0) has already been added

21,907 storage elements found
177,228K bytes of storage allocated

Cached Storage
Addr          Size Key Sp Cblk Time Stamp          Index

Common non-fetch protected storage
12E6DCB0      304 6 241 CFGM B30A8EDF19BD18C3 10
12774310     3056 6 241 CFGM B30A8E3DDBBB1943 10 Index was 29
The control block at 0E289010 (prev: 12B57650) was not available
Unable to locate storage at 0E289010
Cache pointers are in a loop at 12774310 for index 29
The control block at 0E289010 (prev: 12B57730) was not available
Unable to locate storage at 0E289010
2 control blocks found for Common non-fetch protected storage
3376 bytes allocated in Common non-fetch
4366931 total allocations
...
Storage Summary Statistics

Type                                     All
Count      Size      Count      Size
Common Non-fetch protected      21460  177489K      2      3392
Common Fetch protected          369    68488     141    36936
Common persistent                3      192        3      192
Common SCB pool                  80    21128     32    8448
Private Non-fetch protected     492   395848    156   65192
Total                          22571  178149K    334   114160

22599 blocks of storage for 1807728 bytes were obtained to create this report

Analysis of Tcp/Ip for TCPSVT completed

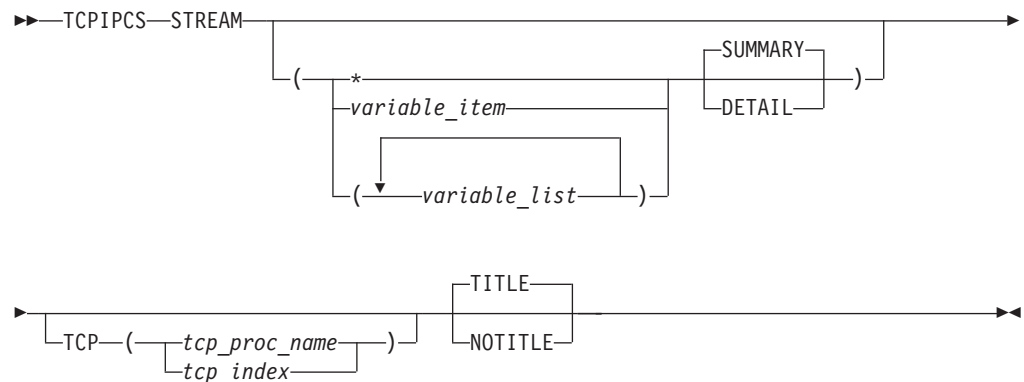
```

TCIPCS STREAM

Invocation of this command displays the stream control blocks.

Syntax

Following is the syntax of the TCIPCS STREAM subcommand:



Parameters

If no parameters are specified, all stream control blocks are summarized.

- * An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space, within parentheses:

Variable parameters are:

CB_address

An address is specified as one to eight hexadecimal digits. An IPCS symbol name may be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string. Displays only the Stream control block associated with one of the following:

SKCB Stream context control block address.

SKQI Stream queue initialization control block address.

SKQP Stream queue pair control block address.

SKQU Stream Queue control block address.

SKSC Stream access control control block address.

SKSH Stream header control block address.

connection_id

Displays the Stream control block with this connection ID. A connection ID is specified as one to eight hexadecimal digits.

In addition to the variable parameters described above, the following keyword parameters may be specified:

SUMMARY

Formats the Stream control blocks in one cross-reference table. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL formats the contents of the Stream control blocks.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIP CS STREAM subcommand:

```
TCPIP CS STREAM
Dataset: IPCS.A594094.DUMPM
Title:   TCPSVT   V2R10: Job(TCPSVT ) EZBITST0(HTCP50A 99.281)+
        00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051
```

The address of the TSAB is: 12E89BB8

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
12E89BF8	1	TCPSVT	V2R10	12B57000	12B570C8	0051	9FFFFFFF	Active

```
1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
```

```

1 TCP/IP(s) for CS      V2R10 found

=====

Analysis of Tcp/Ip for TCPSVT.  Index: 1

TCPIP Stream Analysis

SKRT at 7F78BD88

Sksc@   Sksh@   CID      Driver Api@   Skcb@   Ascb@   Tcb@
7F77E6C8 7F77E7C8 00000007 IP/NAM 00000000 00000000 00000000 00000000
7F70F088 7F61A088 00000006 RAW    00000000 00000000 00000000 00000000
7F70F148 7F61A608 00000005 IP/NAM 00000000 00000000 00000000 00000000
7F70F8C8 7F70F348 00000004 UDP    00000000 00000000 00000000 00000000
7F70F988 7F70FA48 00000003 IP/NAM 00000000 00000000 00000000 00000000
7F78B008 7F7580E8 00000002 TCP    00000000 00000000 00000000 00000000
7F78BCC8 7F78B748 00000001 IP/NAM 00000000 00000000 00000000 00000000

7 Stream(s) found
7 Stream(s) formatted

Analysis of Tcp/Ip for TCPSVT completed

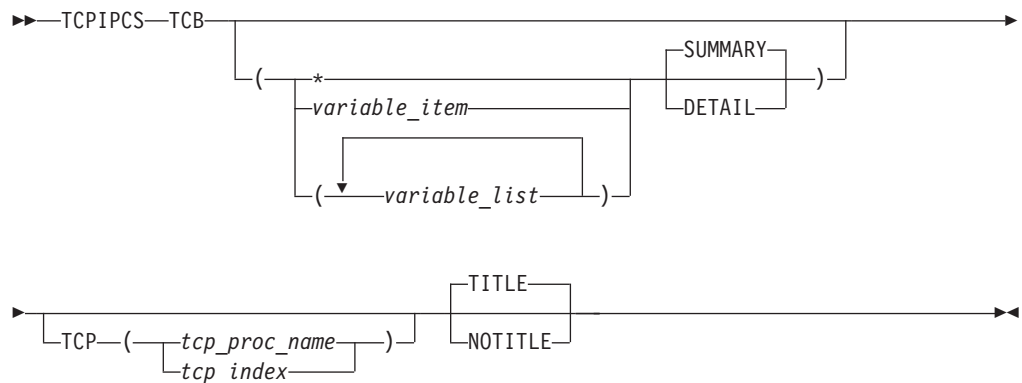
```

TCPIPCS TCB

Invocation of this command displays the Master Transmission Control Block (MTCB) and any Transmission protocol Control Blocks (TCBs) defined in the TCP hash table.

Syntax

Following is the syntax of the TCPIPCS TCB subcommand:



Parameters

If no parameters are specified, all TCP control blocks are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space, within parentheses:

Variable parameters are:

jobname

Displays only the TCBs with this job name. The job name can be a TCP/IP application name or a stack name. A job name is 1–8 alphanumeric characters.

TCB_address

Displays only the TCB with this address. An address is specified as one to eight hexadecimal digits. An IPCS symbol name may be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

connection_id

Displays the TCB with this connection ID. A connection ID is specified as one to eight hexadecimal digits.

In addition to the variable parameters described above, the following keyword parameters may be specified:

SUMMARY

Formats the MTCB contents and lists all the TCBs in one cross-reference table. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL formats the contents of the TCB(s).

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIP CS TCB subcommand:

```
TCPIP CS TCB
Dataset: IPCS.MV21372.DUMPA
Title:   SLIP DUMP ID=TC
```

The address of the TSAB is: 131B8120

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
131B8160	1	TCPSVT	V2R10	13C9F000	13C9F0C8	07D3	94FF755F	Active

1 defined TCP/IP(s) were found

1 active TCP/IP(s) were found

1 TCP/IP(s) for CS V2R10 found

=====

Analysis of Tcp/Ip for TCPSVT. Index: 1

```
TCP/IP Analysis
TCPIP Main TCP Control Block (MTCB)
MTCB: 13C9E890
+0000 M_MAIN_EYE..... TCP MAIN
+0008 M_TCP_LWRITE_Q..... 7F782868
+000C M_TCP_LREAD_Q..... 7F782828
+0014 M_TCP_DRIVER_STATE. 01
+0018 MTCBMTX..... 00000000 00000000 00000000 D7D60601
+0028 MTCBQMX..... 00000000 00000000 00000000 D7D60604
+0038 MTCB_LIST_LOCK..... 00000000 00000000 00000000 D7D60604
+0048 M_PORT_CEILING..... 00000FFF
```

```
+004C M_TPI_SEQ#..... 00000008
+0050 M_PORT_ARRAY..... 7F711FC8
+0054 M_LAST_PORT_NUM.... 0000040C
...
```

TCB	ResrcID	ResrcNm	TcpState	TpiState	Local IPAddr/Port	Remote IPAddr/Port	LuName	App1Name	UserID
7F603108	00000002	TCPSVT	Closed	WLOUNBND	0.0.0.0..0	0.0.0.0..0			
7F605D08	00000017	FTPUNIX1	Listening	WLOIDLE	0.0.0.0..21	0.0.0.0..0			
7F605108	00000013	TCPSVT	Listening	WLOIDLE	0.0.0.0..625	0.0.0.0..0			
7F603508	0000000A	TCPSVT	Listening	WLOIDLE	0.0.0.0..1025	0.0.0.0..0			
7F604508	000000EA	TCPSVT	Established	WLOXFER	197.66.103.1..23	197.11.108.1..1032			
...									
7F607108	0000003E	TCPSVT	Established	WLOXFER	127.0.0.1..1029	127.0.0.1..1028			
7F60A508	000000E8	TCPSVT	Listening	WLOIDLE	0.0.0.0..623	0.0.0.0..0			

25 TCB(s) FOUND
25 TCB(s) FORMATTED

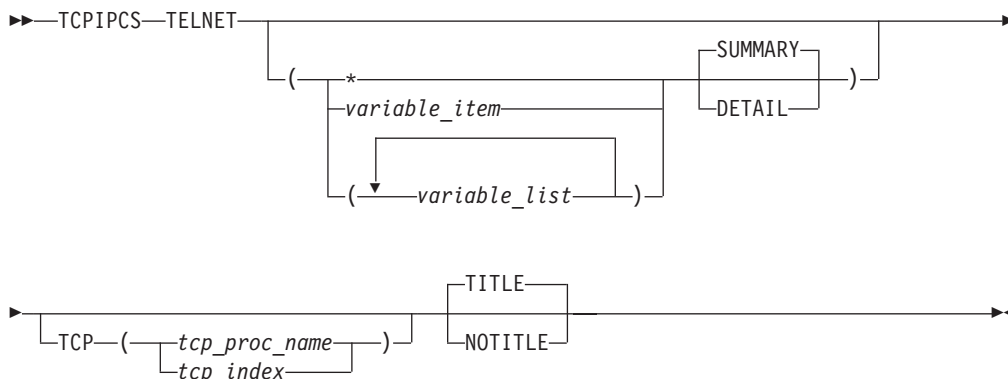
Analysis of Tcp/Ip for TCPSVT completed

TCPIPCS TELNET

Invocation of this subcommand displays either the address, or address and contents, of Telnet control blocks. These include the TCMA, TCFG, TPDB, and, optionally, the TKCB and CVB for a selected session. A partial TCFG that is being built is also displayed, if found.

Syntax

Following is the syntax of the TCPIPCS TELNET subcommand:



Parameters

If no parameters are specified, all TCP control blocks are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space, within parentheses:

Variable parameters are:

LUname

Displays only the session control blocks for the eight-character logical unit name. If the name is less than eight characters, it is padded on the right with blanks.

token Displays only the session control blocks for the token. The token is a 16-digit hexadecimal value. If the token is less than 16 digits, it is padded on the right with zeros.

In addition to the variable parameters described above, the following keyword parameters can be specified:

SUMMARY

Displays the address of the control blocks. SUMMARY is the default.

DETAIL

Displays the contents of the control blocks.

TCP, TITLE, NOTITLE

See "Parameters" on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIP CS TELNET subcommand:

```
TCPIP CS TELNET
Dataset: IPCS.MV21381.DUMPA
Title: SLIP DUMP ID=TC
```

The address of the TSAB is: 13391BC0

Tseab	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
13391C00	1	TCPSVT	V2R10	1323B000	1323B0C8	07DE	04041405	Active
13391C80	2	TCPSVT2	V2R10	00000000	00000000	07E8	00000000	Down Stopping
13391D00	3	TCPSVT1	V2R10	12FC3000	12FC30C8	0080	94FF755F	Active
13391D80	4	TCPSVT3	V2R10	00000000	00000000	0059	00000000	Down Stopping

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

4 TCP/IP(s) for CS V2R10 found

=====

Analysis of Tcp/Ip for TCPSVT. Index: 1

TCPIP Telnet Analysis

TMCA at 7F5B1188

Tpdb@	Port	Tcfg@	Prof	Tkcb@	Token	Cvb@	LUnicode
7F59D8A0	623	7F5A6068	CURR	00000000	00000000	00000000	00000000
7F59D4E0	625	7F59D620	CURR	00000000	00000000	00000000	00000000

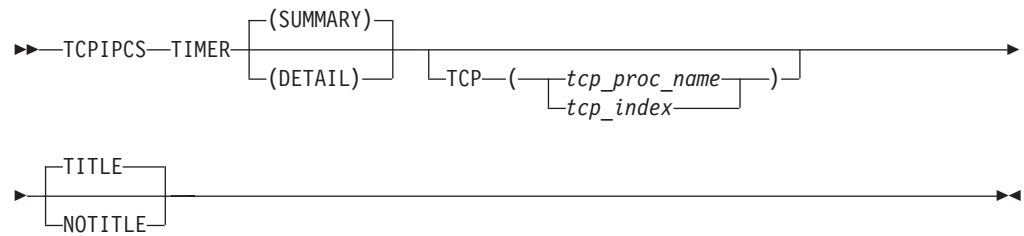
Analysis of Tcp/Ip for TCPSVT completed

TCPIP CS TIMER

Invocation of this command displays the timer control blocks.

Syntax

Following is the syntax of the TCPIP CS TIMER subcommand:



Parameters

Following are the parameters for the TCPIP CS TIMER subcommand:

SUMMARY

Displays the contents of the timer control blocks. The timer queue elements (TQEs) and timer IDs (TIDs) are presented in tabular form.

DETAIL

The timer control blocks are displayed as in the SUMMARY form of the command. In addition, each TQE and each TID is fully displayed.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIP CS TIMER subcommand:

```

TCPIP CS TIMER
Dataset: IPCS.A594094.DUMPF
Title:   CHECK NOT ADDR

The address of the TSAB is: 08CE28C0

Tseb      SI Procedure Version Tsdb      TsdX      Asid TraceOpts Status
08CE2900  1 TCPCS      V2R10    086D8000 086D80C8 01F8 10000100 Active

    1 defined TCP/IP(s) were found
    1 active  TCP/IP(s) were found

    1 TCP/IP(s) for CS      V2R10 found

=====

Analysis of Tcp/Ip for TCPCS.  Index: 1

Timer tables at 086D8F80

ItTmr      Pass      Slot      Delta      Max      PopCount Array@
086D8F80    64        62        100       12800    8253 086D9000

Global TQE Queue for Slot 63:

    Tqe      Tid      Ecb      Mod      Parm      Msec TqeFlag  TidFlag
    08EDDD58 08EDDD44 00000000 EZBIFIU2 08EDDD40    100 00      20

    1 TQE(s) for slot 63 with 0 msec timer offset

ItTmr      Pass      Slot      Delta      Max      PopCount Array@

```

086D8FA0	6	58	1000	128000	825	086D9400
ItTmr	Pass	Slot	Delta	Max	PopCount	Array@
086D8FC0	0	83	10000	1280000	82	086D9800

Global TQE Queue for Slot 122:

Tqe	Tid	Ecb	Mod	Parm	Msec	TqeFlag	TidFLag
086C9020	7F4CEBD0	7F4CEBCC	00000000	00000000	1200000	40	20

1 TQE(s) for slot 122 with 128000 msec timer offset

ItTmr	Pass	Slot	Delta	Max	PopCount	Array@
086D8FE0	0	9	100000	4294967295	8	086D9C00

2 TQE(s) were found

No cancelled TQE(s) were found

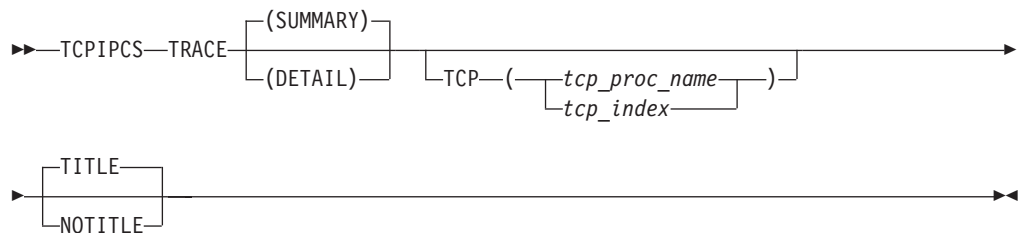
Analysis of Tcp/Ip for TCPCS completed

TCPIPCS TRACE

Invocation of this command displays information about CTrace.

Syntax

Following is the syntax of the TCPIPCS TRACE subcommand:



Parameters

Following are the parameters for the TCPIPCS TRACE subcommand:

SUMMARY

Displays a summary of the CTrace status.

DETAIL

In addition to the SUMMARY information, lists the individual trace buffer entries.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIPCS TRACE subcommand:

```

TCPIPCS TRACE
Dataset: IPCS.R8A0723.RASDUMP2
Title:   EZRPE005
  
```

The address of the TSAB is: 09C445D0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
09C44610	1	TCPCS	V1R4	093C1000	093C10C8	0029	9FFF7E7F	Active
09C44690	2	TCPCS2	V1R4	00000000	00000000	002A	00000000	Down Stopping

2 defined TCP/IP(s) were found
1 active TCP/IP(s) were found

2 TCP/IP(s) for CS V1R4 found

=====

Analysis of Tcp/Ip for TCPCS. Index: 1

Parmlib Member for SYSTCPIP Trace: CTIEZB00
Parmlib Member for SYSTCPIS Trace: CTIIDS00

Trace Control Area

TCA: 092BD410

+0000	TCAACRONYM.....	TCA	
+0006	TCAVERSION.....	0006	
+0008	TCASIZE.....	0000CBD0	
+000C	TCAFTBE.....	092BD7E0	
+0010	TCACURTBE.....	092BE5C8	
+0014	TCACURENT.....	0059C4C0	
+0018	TCATABSZ.....	01000000	
+001C	TCANUMBF.....	00000100	
+0020	TCABUFSZ.....	00010000	
+0024	TCAMXDAT.....	00003800	
+0028	TCAALET.....	01FF000C	
+002C	TCARCNT.....	00004103	
+0030	TCAECNT.....	00004103	
+0034	TCALCNT.....	00000000	
+0038	TCAITOD.....	00000000	00000000
+0040	TCACOMP.....	00000000	
+0044	TCAFLAG.....	03200A80	
+0048	TCAXWRTSEQ.....	00000059	
+004C	TCCTSSWTKN.....	00000000	00000000
+0054	TCAACNT.....	0000	

-- Array elements --

+0058	TCAFILTER_ASID.....	0000
+005A	TCAFILTER_ASID.....	0000
+005C	TCAFILTER_ASID.....	0000
+005E	TCAFILTER_ASID.....	0000
+0060	TCAFILTER_ASID.....	0000
+0062	TCAFILTER_ASID.....	0000
+0064	TCAFILTER_ASID.....	0000
+0066	TCAFILTER_ASID.....	0000
+0068	TCAFILTER_ASID.....	0000
+006A	TCAFILTER_ASID.....	0000
+006C	TCAFILTER_ASID.....	0000
+006E	TCAFILTER_ASID.....	0000
+0070	TCAFILTER_ASID.....	0000
+0072	TCAFILTER_ASID.....	0000
+0074	TCAFILTER_ASID.....	0000
+0076	TCAFILTER_ASID.....	0000

-- End of array --

+0078	TCAUCNT.....	0000
-------	--------------	------

-- Array elements --

+007C	TCAFILTER_USERID....
+0084	TCAFILTER_USERID....
+008C	TCAFILTER_USERID....
+0094	TCAFILTER_USERID....
+009C	TCAFILTER_USERID....

```

+00A4 TCAFILTER_USERID.... .....
+00AC TCAFILTER_USERID.... .....
+00B4 TCAFILTER_USERID.... .....
+00BC TCAFILTER_USERID.... .....
+00C4 TCAFILTER_USERID.... .....
+00CC TCAFILTER_USERID.... .....
+00D4 TCAFILTER_USERID.... .....
+00DC TCAFILTER_USERID.... .....
+00E4 TCAFILTER_USERID.... .....
+00EC TCAFILTER_USERID.... .....
+00F4 TCAFILTER_USERID.... .....
      -- End of array --

+0100 TCAPCNT..... 00000000

      -- Array elements --
+0104 TCAFILTER_PORT..... 0000
+0106 TCAFILTER_PORT..... 0000
+0108 TCAFILTER_PORT..... 0000
+010A TCAFILTER_PORT..... 0000
+010C TCAFILTER_PORT..... 0000
+010E TCAFILTER_PORT..... 0000
+0110 TCAFILTER_PORT..... 0000
+0112 TCAFILTER_PORT..... 0000
+0114 TCAFILTER_PORT..... 0000
+0116 TCAFILTER_PORT..... 0000
+0118 TCAFILTER_PORT..... 0000
+011A TCAFILTER_PORT..... 0000
+011C TCAFILTER_PORT..... 0000
+011E TCAFILTER_PORT..... 0000
+0120 TCAFILTER_PORT..... 0000
+0122 TCAFILTER_PORT..... 0000
      -- End of array --

+0124 TCAICNT..... 00000000

      -- Array elements --
+0128 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0138 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+0148 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0158 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+0168 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0178 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+0188 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0198 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+01A8 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+01B8 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+01C8 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+01D8 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+01E8 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+01F8 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+0208 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0218 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+0228 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0238 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+0248 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0258 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+0268 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0278 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+0288 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0298 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+02A8 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+02B8 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+02C8 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+02D8 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
+02E8 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+02F8 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000

```

```

+0308 TCAFILTER_IPADDRESS. 00000000 00000000 00000000 00000000
+0318 TCAFILTER_IPSUBMASK. 00000000 00000000 00000000 00000000
      -- End of array --

+0330 TCAWRAPTIME..... B66479D0 DE1B3402
+0338 TCAWRAPCOUNT..... 00000001
+033C TCAXWTRCNT..... 00000000
+0340 TCACURCUR..... 092BFFE0
+0344 TCANXTCUR..... 01004E80
+0348 TCADATSZ..... 02000000
+034C TCADATBF..... 00000200
+0350 TCAWCONT..... 00000068
+0354 TCATRCNT..... 00000068
+0358 TCALSCNT..... 00000000
+035C TCASEQXWRT..... 00000001
+0360 TCAPTSSWTKN..... 02895060 00000001
+0368 TCAISTBE..... 092C4FE0
+036C TCAISNRTBE..... 00000200
+0370 TCAISBUFSZ..... 00010000
+0374 TCAISTBLSZ..... 02000000
+0378 TCAISTRcnt..... 00000000
+037C TCAISWRCNT..... 00000000
+0380 TCAISLSCNT..... 00000000
+0384 TCAISROCNT..... 00000000
+0388 TCAISXWSEQ..... 00000001
+0390 TCAISCDS..... 092C4FE0 03001000
+0390 TCAISCDTBE..... 092C4FE0
+0394 TCAISCDBUF..... 03001000
+0398 TCAISXWTKN..... 00000000 00000000
+03A0 TCAISWRTIM..... 00000000 00000000
+03A8 TCAISLSTIM..... 00000000 00000000
+03B0 TCADUMPSZ..... 00000000
+03B4 TCADUMPDS..... 00000000
+03B8 TCADUMPOF..... 00000000
+03C0 TCADUMPTOD..... 00000000 00000000

```

Event Trace Statistics for SYSTCPIP

```

Size of the Trace Control Area . . . . 52176
Size of the trace buffer . . . . . 16384K
Size of a trace segment. . . . . 64K
Number of trace segments . . . . . 256
Maximum trace record size. . . . . 14,336
Number of trace records requested. . . 16,643
Number of trace records recorded . . 16,643
Number of trace segments filled. . . 89
Average records per segment. . . . . 187
Average records per table. . . . . 47,872
Trace status . . . . . Active
XWriter status . . . . . Disconnected
Number of buffers written. . . . . 0
Lost record count. . . . . 0
Lost record time . . . . . 1900/01/01 00:00:00.000000
Trace table wrap count . . . . . 1
Trace table wrap time. . . . . 2001/09/05 12:41:47.461043
Average records per wrap . . . . . 16,643

```

Data Trace Statistics for SYSTCPDA

```

Size of the trace buffer . . . . . 32768K
Size of a trace segment. . . . . 64K
Number of trace segments . . . . . 512
Number of trace records requested. . . 104
Number of trace records recorded . . 104
Number of trace segments filled. . . 1
Trace status . . . . . Active
XWriter status . . . . . Connected
Number of lost records . . . . . 0

```

```

IDSTRACE Statistics for SYSTCPIS
Size of the trace buffer . . . . . 32768K
Size of a trace segment. . . . . 64K
Number of trace segments . . . . . 512
Number of trace records requested. . . 0
Number of trace records recorded . . . 0
Number of trace segments filled. . . . 1
Trace status . . . . . Active
XWriter status . . . . . Disconnected
Number of lost records . . . . . 0
Lost record time . . . . . 1900/01/01 00:00:00.000000
Trace table wrap count . . . . . 0
Trace table wrap time. . . . . 1900/01/01 00:00:00.000000

Tseb_Trace_Opts: 9FFF7E7F
Options: Init Opcmds Opmsgs Socket AFP XCF Access PFS API
         Engine Streams Queue RAW UDP TCP ICMP ARP CLAW LCS
         Internet Message WorkUnit Config SNMP IOCTL FireWall
         VtamData TelnVtam Telnet Vtam

256 SYSTCPIP Trace Buffer Elements were found
 0 SYSTCPIP Trace Buffer Elements were formatted
512 SYSTCPDA Trace Buffer Elements were found
 0 SYSTCPDA Trace Buffer Elements were formatted
512 SYSTCPIS Trace Buffer Elements were found
 0 SYSTCPIS Trace Buffer Elements were formatted

```

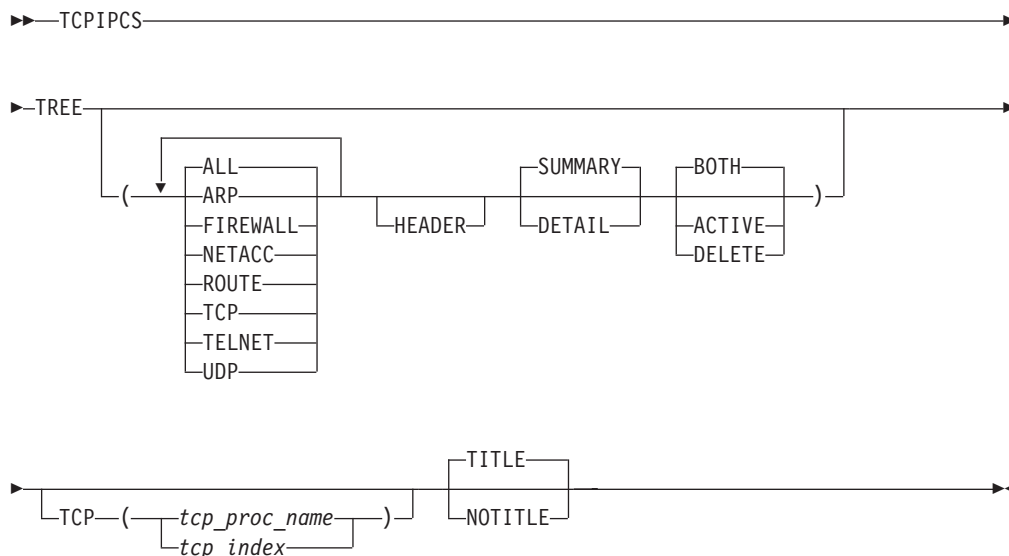
Analysis of Tcp/Ip for TCPCS completed

TCPIPCS TREE

Invocation of this command displays the structure of TCP/IP Patricia trees.

Syntax

Following is the syntax of the TCPIPCS TREE subcommand:



Parameters

The following keyword parameters may be specified:

ALL

Display structure of all TCP/IP trees. ALL is the default.

ARP

Only display structure of ARP trees.

FIREWALL

Only display structure of Firewall trees.

IQDIO

Only display structure of iQDIO trees.

ND

Only display structure of Neighbor Discovery trees.

NETACC

Only display structure of NetAccess trees.

ROUTE

Only display structure of both IPv4 and IPv6 route trees.

ROUTEV4

Only display structure of IPv4 route trees.

ROUTEV6

Only display structure of IPv6 route trees.

TCP

Only display structure of TCP trees.

TELNET

Only display structure of Telnet trees.

HEADER

Display tree header information. Not displayed by default.

SUMMARY

Display the addresses of the control blocks and other data in trees. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL also shows the search key values.

BOTH

Display both active and logically deleted tree nodes. BOTH is the default.

ACTIVE

Only display active tree nodes.

DELETE

Only display logically deleted tree nodes

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {ALL ARP, FIREWALL, IQDIO, ND, NETACC, ROUTE, ROUTEV4, ROUTEV6, TCP, TELNET}, all of them will be used.
2. If you specify multiple keywords from the set {BOTH, ACTIVE, DELETE}, only the last one will be used.
3. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCIPCS TREE subcommand:

```

TCPIP CS TREE
Dataset: IPCS.R8A0723.RASDUMP
Title: EZRPE005
The address of the TSAB is: 098221F0
Tseb SI Procedure Version Tsd Asid TraceOpts Status
09822230 1 TCPCS V1R4 08E85000 08E850C8 001E 9FFF7E7F Active
098222B0 2 TCPCS2 V1R4 08937000 089370C8 01F6 9FFF7E7F Active
2 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
2 TCP/IP(s) for CS V1R4 found
=====
Analysis of Tcp/Ip for TCPCS. Index: 1
TCPIP Tree Analysis
ARP Tree for PIF at 7F503B88
Node@ Bit Parent LChild RChild Key Element Status
7F1F42B8 0 00000000 7F1E3FF8 7F1E3FD8 08E85E00 00000000 Deleted
7F1E3FF8 4 7F1F42B8 7F1F42B8 7F1E3FF8 7F1F1C20 7F1C0A48 Active
7F1E3FD8 0 7F1F42B8 7F1F42B8 7F1E3FD8 7F1F1C18 7F1F41E8 Active
2 active elements in ARP Tree
ARP Tree for PIF at 7F503F08
7F1E3278 0 00000000 7F1E4038 7F1E4018 08E85E00 00000000 Deleted
7F1E4038 7 7F1E3278 7F1E3278 7F1E4038 7F1F1C30 7F1A8AE8 Active
7F1E4018 0 7F1E3278 7F1E3278 7F1E4018 7F1F1C28 7F1A8A48 Deleted
1 active elements in ARP Tree
ARP Tree for PIF at 7F504288
7F1F41B8 0 00000000 7F1E4078 7F1E4058 08E85E00 00000000 Deleted
7F1E4078 3 7F1F41B8 7F1F41B8 7F1E4078 7F1F1C40 7F1A8B88 Active
7F1E4058 0 7F1F41B8 7F1F41B8 7F1E4058 7F1F1C38 7F1A8A48 Deleted
1 active elements in ARP Tree
ND Tree for PIF at 7F503F08
7F1E4038 7 7F1E3278 7F1E3278 7F1E4038 7F1F1C30 7F1A8AE8 Active
1 active elements in ND Tree

Route IPv4 Search Tree
Node@ Bit Parent LChild RChild Key Rtp@ Element
7F66AF68 255 00000000 7F3F6028 7F66A268
7F50B0A8 25 7F207068 7F66AF68 7F1EF1A8 7F1EF0A8 7F1EEFA8 7F1EEE28
7F207068 28 7F52C008 7F50B0A8 7F1EF4C8 7F2070A8 7F301008 7F3F6068
7F3F6028 32 7F66AF68 7F66AF68 7F3F6028 7F50B1A8 7F50B0E8 7F621348
4 elements in Route IPv4 Search Tree

Route IPv6 Search Tree
Node@ Bit Parent LChild RChild Key Rtp@ Element
7F66AF68 255 00000000 7F3F6028 7F66A268
7F50B0A8 25 7F207068 7F66AF68 7F1EF1A8 7F1EF0A8 7F1EEFA8 7F1EEE28
7F207068 28 7F52C008 7F50B0A8 7F1EF4C8 7F2070A8 7F301008 7F3F6068
7F52C008 29 7F66A268 7F207068 7F1EEC28 7F1EEB88 7F1EEAC8 7F1EE948
4 elements in Route IPv6 Search Tree

Route IPv4 Update Tree
Node@ Bit Parent LChild RChild Key Rtp@ Element
7F66AF08 255 00000000 7F50B208 7F66A2A8
7F1EECA8 3 7F6210E8 7F1EECA8 7F6210E8 7F621028 7F6860A8 7F69B130
7F1EF8E8 7 7F207028 7F207028 7F1EF8E8 7F1EF3C8 7F1EF368 7F1EF1E8
7F1EF108 3 7F50B028 7F1EF108 7F50B028 7F52C048 7F6212E8 7F621168
7F50B028 25 7F207028 7F66AF08 7F1EF108 7F1EF048 7F1EEFA8 7F1EEE28
4 elements in Route IPv4 Update Tree

Route IPv6 Update Tree
Node@ Bit Parent LChild RChild Key Rtp@ Element
7F66AF08 255 00000000 7F50B208 7F66A2A8
7F1EECA8 3 7F6210E8 7F1EECA8 7F6210E8 7F621028 7F6860A8 7F69B130
7F1EF8E8 7 7F207028 7F207028 7F1EF8E8 7F1EF3C8 7F1EF368 7F1EF1E8
7F50B028 25 7F207028 7F66AF08 7F1EF108 7F1EF048 7F1EEFA8 7F1EEE28
4 elements in Route IPv6 Update Tree

Analysis of Tcp/Ip for TCPCS completed

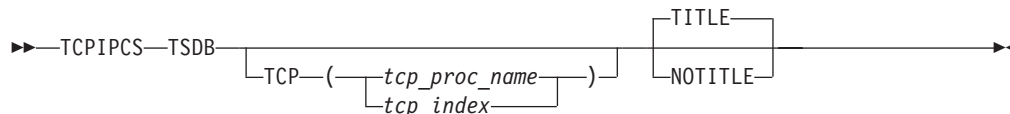
```

TCIPCS TSDB

Invocation of this subcommand displays the TSDB server data block.

Syntax

Following is the syntax of the TCIPCS TSDB subcommand:



Parameters

Following are the parameters for the TCIPCS TSDB subcommand:

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Sample output

The following is sample output of the TCIPCS TSDB subcommand:

```
TCIPCS TSDB
Dataset: IPCS.MV21381.DUMPA
Title:   SLIP DUMP ID=TC
```

The address of the TSAB is: 13391BC0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
13391C00	1	TCPSVT	V2R10	1323B000	1323B0C8	07DE	04041405	Active
13391C80	2	TCPSVT2	V2R10	00000000	00000000	07E8	00000000	Down Stopping
13391D00	3	TCPSVT1	V2R10	12FC3000	12FC30C8	0080	94FF755F	Active
13391D80	4	TCPSVT3	V2R10	00000000	00000000	0059	00000000	Down Stopping

```
4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
```

```
4 TCP/IP(s) for CS      V2R10 found
```

```
=====
Analysis of Tcp/Ip for TCPSVT.  Index: 1
```

TSDB control block summary

```
TSDB: 1323B000
+0000 TSDB_ACRONYM..... TSDB
+0004 TSDB_LENGTH..... 00C8
+0006 TSDB_VERSION..... 0003
+0008 TSDB_STATE..... 0015
+000A TSDB_ASID..... 07DE

-- Array elements --
+0010 TSDB_MT..... 11A7E870
+0014 TSDB_MT..... 962F5E00
....
+0060 TSDB_CTRACE_PARMLIB_NAME. CTIEZB02
+006C TSDB_SMCA..... 00000000
+0070 TSDB_TSRMT..... 00000000
+0074 TSDB_FLAGS..... 00000000
+0078 TSDB_CONFIG_PORT..... 00000401
+007C TSDB_OSASF_PORT..... FFFFFFFF
+0080 TSDB_EZBITMSN@..... 91A8BF90
+0084 TSDB_TERMINATING_ECB..... 807EC758
```

```

+0088 TSDB_DUAF..... 00000000
+008C TSDB_TSCA..... 13236A58
+0090 TSDB_SOCIFPTR..... 91BC3E78
+0094 TSDB_SOMIFPTR..... 91BCA050
+0098 TSDB_RXGLUPTR..... 91BF6308
+009C TSDB_FFSTADDR..... 80B46E18
+00A0 TSDB_FFST_PHMSGTIME..... 00000000 00000000
+00A8 TSDB_LEPARMS..... 14B01BBA
+00AC TSDB_OE_AS_STOKEN..... 00000038 00000001
+00B4 TSDB_SOMT2..... 91C3FE60

```

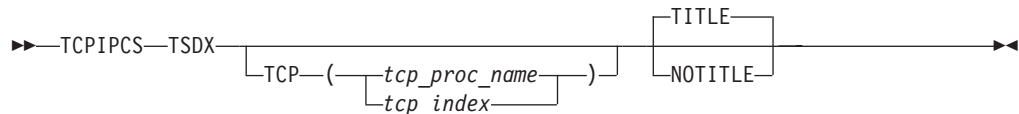
Analysis of Tcp/Ip for TCPSVT completed

TCPIPCS TSDX

Invocation of this subcommand displays the TSDX server data extension.

Syntax

Following is the syntax of the TCPIPCS TSDX subcommand:



Parameters

Following are the parameters for the TCPIPCS TSDX subcommand:

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Sample output

The following is sample output of the TCPIPCS TSDX subcommand:

```

TCPIPCS TSDX
Dataset: IPCS.MV21381.DUMPA
Title: SLIP DUMP ID=TC

```

The address of the TSAB is: 13391BC0

Tseb	SI	Procedure	Version	Tsdb	TsdX	Asid	TraceOpts	Status
13391C00	1	TCPSVT	V2R10	1323B000	1323B0C8	07DE	04041405	Active
13391C80	2	TCPSVT2	V2R10	00000000	00000000	07E8	00000000	Down Stopping
13391D00	3	TCPSVT1	V2R10	12FC3000	12FC30C8	0080	94FF755F	Active
13391D80	4	TCPSVT3	V2R10	00000000	00000000	0059	00000000	Down Stopping

```

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

```

```

4 TCP/IP(s) for CS V2R10 found

```

=====

Analysis of Tcp/Ip for TCPSVT. Index: 1

TSDX control block summary

```

TSDX: 1323B0C8
+0000 TSDX_ACRONYM..... TSDX
+0004 TSDX_LENGTH..... 0300
+0006 TSDX_VERSION..... 0003
+0008 TSDX_FLAGS..... 60000001
+000C TSDX_ASCB..... 00F7C280

```



```

+0010 TSDX_PROCNAME..... TCPSVT
+0018 TSDX_CART..... 00000000 00000000
+0020 TSDX_CONSID..... 00000001
+0024 TSDX_TCB..... 007EC9A8
+0028 TSDX_TCB_TOKEN..... 00001F78 00000008 00000003 007EC9A8
+0038 TSDX_TCPIP_DS_ALET..... 01FF0011
+003C TSDX_TCPIP_DS_ADDR..... 00001000
+0040 TSDX_TCPIP_DS_END..... 19001000
+0044 TSDX_ET_TOKEN..... 7FFD9D10
...
+026C TSDX_CSMSTATAREA..... 141C7A88
+0270 TSDX_CSMDUMPINFO..... 141C7A90
+0288 TSDX_AUTOLOG_TASK_ECB..... 807EC758
+028C TSDX_AUTOLOG_CB..... 1333C0A8
+0290 TSDX_SASTRT_ECB..... 807EC758
+0294 TSDX_XFCVT..... 13096410
+0298 TSDX_XCFLOCK..... 00000000 00000000 00000000 D7D60901

```

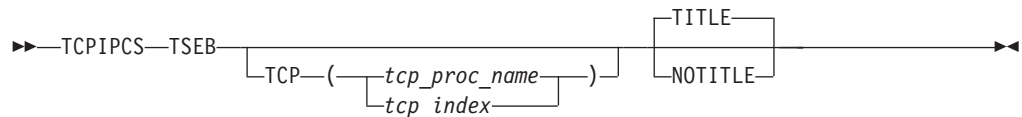
Analysis of Tcp/Ip for TCPSVT completed

TCPIPCS TSEB

Invocation of this subcommand displays the TSEB server anchor block.

Syntax

Following is the syntax of the TCPIPCS TSEB subcommand:



Parameters

Following are the parameters for the TCPIPCS TSEB subcommand:

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Sample output

The following is sample output of the TCPIPCS TSEB subcommand:

```

TCPIPCS TSEB
Dataset: IPCS.MV21381.DUMPA
Title:  SLIP DUMP ID=TC

```

The address of the TSAB is: 13391BC0

Tseb	SI	Procedure	Version	Tsdb	Tsdx	Asid	TraceOpts	Status
13391C00	1	TCPSVT	V2R10	1323B000	1323B0C8	07DE	04041405	Active
13391C80	2	TCPSVT2	V2R10	00000000	00000000	07E8	00000000	Down Stopping
13391D00	3	TCPSVT1	V2R10	12FC3000	12FC30C8	0080	94FF755F	Active
13391D80	4	TCPSVT3	V2R10	00000000	00000000	0059	00000000	Down Stopping

```

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

```

```

4 TCP/IP(s) for CS      V2R10 found

```

=====

Analysis of Tcp/Ip for TCPSVT. Index: 1

TSEB control block summary

```

TSEB: 13391C00
+0000 TSEB_ACRONYM..... TSEB
+0004 TSEB_LENGTH..... 0080
+0006 TSEB_VERSION..... 0003
+0008 TSEB_FLAGS..... 82000000
+0008 TSEB_STATUS..... 82
+000C TSEB_REQUESTORS..... 00000000
+0010 TSEB_TCPIP_NAME..... TCPSVT
+0018 TSEB_SI..... 01
+0019 TSEB_IID..... 04
+001A TSEB_TCPIP_VERSION... 0510
+001C TSEB_TSDB..... 1323B000
+0020 TSEB_LX..... 00002E00
+0024 TSEB_TCA..... 11469E50
+0028 TSEB_TRACE_OPTS..... 04041405
+002C TSEB_TRACE_OPT2..... 00000000
+0034 TSEB_SCHEDULED_EVENTS. 00000000
+0038 TSEB_ASID..... 07DE
+003C TSEB_LPA_SADDR..... 11A719E0
+0040 TSEB_LPA_EADDR..... 11C62FFF
+0044 TSEB_QDIO_BGRP_Q@.... 1320A648
+0048 TSEB_EZBITDCR..... 9320ACF8
+004C TSEB_ITCVT..... 1323B3C8
+0050 TSEB_BGRP_Q@..... 1320A608
+0054 TSEB_DUAF..... 130A4010
+0059 TSEB_TOKENID..... 000015
+005C TSEB_TCMTPTR..... 132072F0
+0060 TSEB_EZBITCOM_LEN.... 00007D28
+0064 TSEB_CS390_VERSION... 020A
+0068 TSEB_CSMFREE..... 1320A548
+006C TSEB_TCPIP_STOKEN.... 00001F78 00000008
+0074 TSEB_CSMPACK..... 1320A588
+0078 TSEB_SOCA..... 140BAEB8
+007C TSEB_CSMPACKQDIO..... 1320A5C8

```

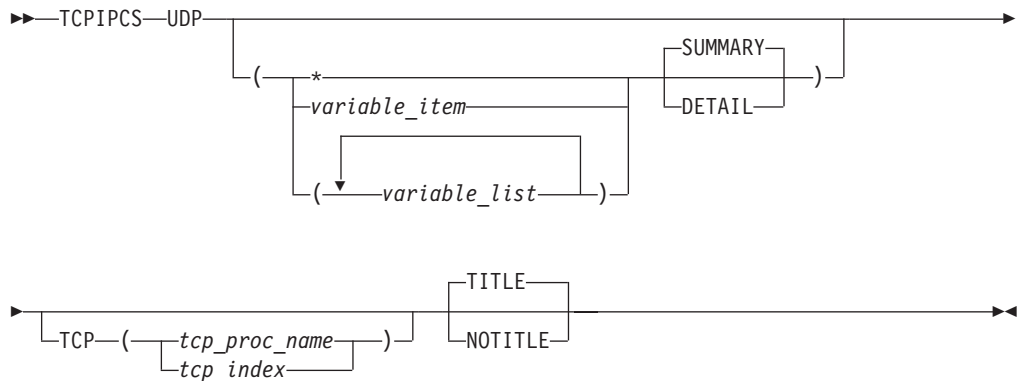
Analysis of Tcp/Ip for TCPSVT completed

TCPIPCS UDP

Invocation of this command displays the Master UDP Control Block (MUCB) and any UDP Control Blocks (UCBs) in the UDP hash tables or link list.

Syntax

Following is the syntax of the TCPIPCS UDP subcommand:



Parameters

If no parameters are specified, all UDP control blocks are summarized. Following are the parameters for the TCPIPCS UDP subcommand:

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space, within parentheses:

Variable parameters are:

jobname

Displays only the UDP control blocks with this job name. The job name may be a TCP/IP application name or a stack name. A job name is 1–8 alphanumeric characters.

UCB_address

Displays only the UDP control block with this address. An address is specified as one to eight hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

connection_id

Displays the UDP control block with this connection ID. A connection ID is specified as 1–8 hexadecimal digits.

In addition to the variable parameters described above, the following keyword parameters can be specified:

SUMMARY

Formats the MUCB contents and lists all the UDPs in one cross-reference table. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL formats the contents of the UCBs.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIPCS UDP subcommand:

```
| TCPIPCS UDP
| Dataset: IPCS.R8A0723.RASDUMP
| Title: EZRPE005
| The address of the TSAB is: 098221F0
| Tseb SI Procedure Version TsdB TsdX Asid TraceOpts Status
| 09822230 1 TCPCS V1R4 08E85000 08E850C8 001E 9FFF7E7F Active
| 098222B0 2 TCPCS2 V1R4 08937000 089370C8 01F6 9FFF7E7F Active
| 2 defined TCP/IP(s) were found
| 2 active TCP/IP(s) were found
| 2 TCP/IP(s) for CS V1R4 found
|
| Analysis of Tcp/Ip for TCPCS. Index: 1
| User Datagram Protocol Control Block Summary
| MUCB: 7F50B248
| +0000 UMUCBEYE. MUCB USTKLNKD. 01 UDRVSTAT. 00
| +0008 UMUCB6FLG..... 00010000
| +0009 U6STKLNKD..... 01
| +000B U6DRVSTAT..... 00
```

```

+000C UOPENPRT. 00000000 UFREEPRT. 041C MCBMUTEX. 00000000
      00000000 00000000 D7D60402
+0028 UDPCFG... 00000001 0000FFFF 0000FFFF 00000001 80000000
      00000000
+0040 UDPCFG2.. 00000001 0000FFFF 0000FFFF 00000001 80000000
      00000000
+0058 UDPMIB... 00000008 0000004B 00000000 0000004D
      USBCAST.. 00000000 USLPBACK. 00000000 USDN
+0074 USRCVBUF. 0000FFFF USSNDBUF. 0000FFFF UFGPRC... 00
      SERIALV. 00000003 SERIAL1. 00000000 ULAS
+008C ULASTPRT. 0000 ULASTUCB. 00000000 SERIAL2. 00000000
      UIPWRQ@.. 7F407968 UIPRDQ@.. 7F407928 UIP6
+00A4 UIP6RDQ@. 7F207928
+00BC UDMULTI_NUM..... 00000000
+00C0 UDMUX_TOKEN..... 7F407B88 00000003
+00D0 UDMULTI@. 00000000
+00D4 UD6MULTI_NUM..... 00000000
+00D8 UD6MULTI@..... 00000000
+00DC UD6MUX_TOKEN..... 7F207B88 00000006
+00E4 UD6MUX_MULTI_TOKEN. 00000000 00000000

```

IPv6 Unicast Hash Table

```

UCB      ResrcID ResrcNm TpiState Port IPAddr
7F2FCD00 0000000C TCPCS WLOUNBND  :0
1 UCB(s) FOUND
1 UCB(s) FORMATTED

```

IPv4 Unicast Hash Table

```

UCB      ResrcID ResrcNm TpiState Port IPAddr
7F50C000 00000004 TCPCS WLOUNBND  0.0.0.0
1 UCB(s) FOUND
1 UCB(s) FORMATTED

```

IPv6 Multicast Hash Table

```

0 UCB(s) FOUND
0 UCB(s) FORMATTED

```

IPv4 Multicast Link List

```

0 UCB(s) FOUND
0 UCB(s) FORMATTED

```

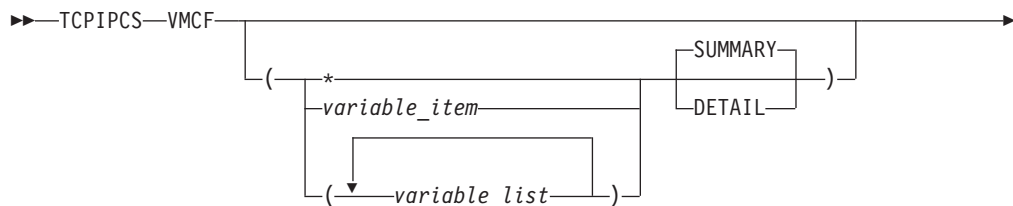
Analysis of Tcp/Ip for TCPCS completed

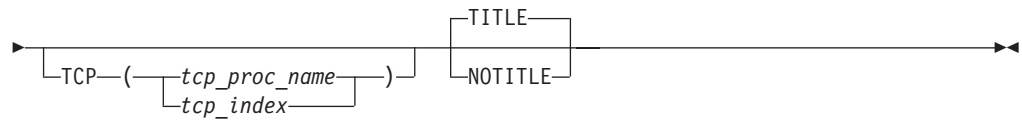
TCIPCS VMCF

Invocation of this command displays information about VMCF (Virtual Machine Communication Facility) and IUCV (Inter-User Communication Vehicle) users.

Syntax

Following is the syntax of the TCIPCS VMCF subcommand:





Parameters

If no parameters are specified, all VMCF control blocks are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

From 1–32 of the following variable parameters can be repeated, each separated by a blank space, within parentheses:

Variable parameters are:

user_id

Displays only the VMCF control block associated with this user ID. Specified as one to eight alphanumeric characters.

ASCB_address

Displays only the VMCF control blocks associated with this address space control block address. An address is specified as one to eight hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

ASID_number

Displays only the VMCF control blocks associated with this address space identifier. An ASID is specified as one to four hexadecimal digits.

In addition to the variable parameters described above, the following keyword parameters can be specified:

SUMMARY

Formats the VMCF control blocks in one cross-reference table. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL formats the contents of selected VMCF USER control blocks.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIPICS VMCF subcommand:

```
TCPIPICS VMCF ((* ) SUMMARY)
Dataset: IPCS.JW11111.DUMPA
Title:   IPCS VMCF DUMP
```

The address of the TSAB is: 08EBC180

```
Tseb      SI Procedure Version TsdB      TsdX      Asid TraceOpts Status
```

```
08EBC1C0 1 TCPCS      V2R10   089DC000 089DC0C8 01F7 9FFFFFF7F Active
```

```
1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
```

```
1 TCP/IP(s) for CS      V2R10 found
```

```
=====
```

```
Analysis of Tcp/Ip for TCPCS. Index: 1
```

```
TCPIP VMCF Analysis
```

```
XINF at 09813000
```

```
VMCF CVT          : 00A44078
User Array        : 09813090
Userid Count      : 1
Userid Array      : 09817050
Userid            : VMCF
MSGBUILD          : 89802838
MVPMSGs           : 8981A290
Ecb               : 00000000
TNF CVT           : 00A63808
VMCF QD           : 00000000
VMCF QD Count     : 0
TNF Manager Area  : 00008FE0
SMSG Id           : 0
```

```
USER at 09813C50
```

```
Userid           : USER18
Asid              : 005D
No UserData
```

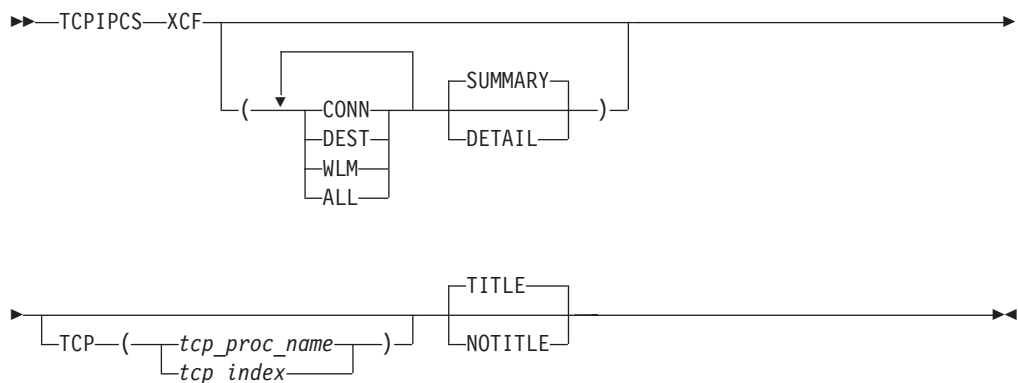
```
Analysis of Tcp/Ip for TCPCS completed
```

TCPIPCS XCF

Invocation of this command produces a cross-system coupling facility (XCF) analysis report.

Syntax

Following is the syntax of the TCPIPCS XCF subcommand:



Parameters

If no parameters are specified, the dynamic VIPA hash table and partner tables are summarized.

The following keyword parameters can be specified:

CONN

Only display connection hash table optional information. CONN is the default.

DEST

Only display destination hash table optional information.

WLM

Only display workload manager optional information.

ALL

Display all optional information.

SUMMARY

Formats the XCF control blocks in one cross-reference table. SUMMARY is the default.

DETAIL

In addition to the SUMMARY display, DETAIL formats the contents of XCF control blocks.

TCP, TITLE, NOTITLE

See “Parameters” on page 171 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {ALL, CONN, DEST, WLM}, all of them will be used.
2. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample output

The following is sample output of the TCPIP XCF subcommand:

```
TCPIP XCF
Dataset: IPCS.MV20603.DUMPA
...
----XFCVT information----

XFCVT@      12CC7410      Member Name RUSSIATCPSVT
Local PTB    12CC752C      PTB Chain  1276A410
DVIPAHASH@   13239408      IPHashT@   12A9C010
ConnRteHash@ 12A9B010      DPTHASH@   1277D010
WLMDATA@     00000000      PolicyPart@ 7F635108
=====
----DVIPA Hash Table----
DVIPA Hash Table at 13239408
Hash table has size 2056 bytes

DVIPA address      197.11.200.2      index 3
MVSName/TCPName                                Status/Rank

                RUSSIA/TCPSVT                32/255
                GERMANY/TCPSVT                12/0
...
Found 9 entries in the DVIPA Hash Table.
=====
=====
Local Partner Table
=====
----Partner Table Control Block----
Partner Table at 12CC752C
NextPtr: 00000000
MVSName:  RUSSIA                CPName:    RUSSIA
TCPName:  TCPSVT                IPTable:   12D6F140
IPCount:  21                    IPEntries@: 1322D0E8
...
----Dynamic VIPA Table----
```

```

Sending Partner0: 128E1410 GERMANY/TCPSPVT

Current Dynamic Home Address: 199.11.87.104
Table Address: 12A98C10 Table Length: 8208
Number of Table Entries: 7
.....
DVIPA entry at 12A98C40
  DVIPA origin: DEFINE Dist Status: Unknown:0
  DVIPA Flags: MoveImmed
  DVIPA Flag2: ()
  IP address: 197.11.104.10 Mask: 255.255.255.0
...
=====
Next Partner Table
=====
----Partner Table Control Block----
Partner Table at 1276A410
NextPtr: 12659410
MVSName: SPAIN CPName: SPAIN
TCPName: TCPSVT IPTable: 13BA63A0
...

```

ERRNO

The ERRNO command searches for the name and description of constants used for ERRNO, ErrnoJr, module ID, reason code, and ABEND reason code.

Syntax

Following is the syntax of the ERRNO command:



Parameters

Following are the parameters of the ERRNO command:

type

The optional type of value provided:

A Abend code

E Errno

J ErrnoJr

M	Module ID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

R Reason code (default)

value

The decimal or hexadecimal value to be converted. By default, the value is assumed to be a hexadecimal number. If the value is less than the maximum size for its type, the value is padded on the left with zeros. Choices are:

hhhhhhhhh

An address consisting of one to eight hexadecimal digits ending with a period. The value at that address will be interpreted.

hhhhhhhh

An ERRNO, ERRNO junior, reason code, ABEND code, or module ID consisting of one to eight hexadecimal digits.

hhhhhhhhx

An ERRNO, ERRNO junior, or a module ID consisting of one to eight hexadecimal digits followed by the letter x.

dddddddn

An ERRNO, ERRNO junior, or a module ID consisting of one to eight decimal digits followed by the letter n.

name

The name of a module, an ERRNO, an ErrnoJr, or an ABEND reason code.

Note: If the name is not found, ERRNO attempts to interpret the name as a hexadecimal value.

Sample output

Following are some sample outputs of the ERRNO command.

- **Example 1:** Reason code by hexadecimal value.

```
Command ==> errno r 74be72e9
```

```
ReasonCode: 74BE72E9
Module: EZBITSTO ErrnoJr: 29417 JRCMNOCSM
Description: Cache Manager encountered a CSM storage shortage
```

- **Example 2:** Reason code by address, where the value at address 07093F98 is 74717273. Note that type R (reason code) is the default.

```
Command ==> errno 7093f98.
```

```
ReasonCode: 74717273
Module: EZBPFWRT ErrnoJr: 29299 JRARPSVNOTDEFINED
Description: The ATMARPSV name specified is not defined
```

- **Example 3:** Errno in decimal.

```
Command ==> errno e 129n
```

```
Errno: 00000081(129) : ENOENT
Description: No such file, directory, or IPC member exists
```

- **Example 4:** ErrnoJr in hexadecimal.

```
Command ==> errno j 6c
```

```
ErrnoJr: 0000006C(108) : JRFILENOTTHERE
Description: The requested file does not exist
```

- **Example 5:** Abend code in decimal.

```
Command ==> errno a 9473n
```

```
Abend Reason Code: 00002501
Module: Unknown Reason: TcpiStorNoCSMstorage
Description: No CSM storage available
```

- **Example 6:** Module ID in hexadecimal.

```
Command ==> errno m 74be
```

```
ModuleId: 74BE(29886) : EZBITSTO EZBTIINI
```

- **Example 7:** Module name.

```
Command ==> errno ezbifinb
```

```
ModuleId: 7418(29720) : EZBIFINB EZBTIINI
```

- **Example 8:** ERRNO name.

```
Command ==> errno ebadf
```

```
Errno: 00000071(113) : EBADF
```

```
Description: The file descriptor is incorrect
```

- **Example 9:** ErrnoJr name.

```
Command ==> errno jrmaxuids
```

```
ErrnoJr: 00000013(19) : JRMAXUIDS
```

```
Description: The maximum number of OpenMVS user IDs is exceeded
```

- **Example 10:** ABEND reason name.

```
Command ==> errno tcpbadentrycode
```

```
Abend Reason Code: 00000401
```

```
Module: Unknown Reason: TcpBadEntryCode
```

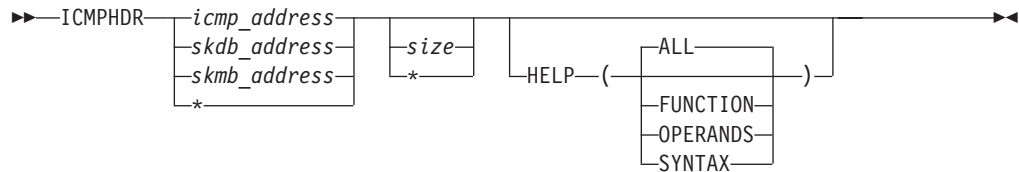
```
Description: Bad Entry code to module
```

ICMPHDR

Invocation of the ICMPHDR command displays the ICMP header fields.

Syntax

Following is the syntax of the ICMPHDR command:



Parameters

Following are the parameters of the ICMPHDR subcommand:

- * To omit this positional parameter when using the HELP keyword.

icmp_address

The address of an ICMP header or the symbol for the address.

skdb_address

The address of an SKDB control block or the symbol for the address.

skmb_address

The address of an SKMB control block or the symbol for the address.

size

The amount of data to display. If the size is greater than the size of the header, the variable portion of the header is displayed if it exists. Must be one to three hexadecimal digits.

HELP

Display IPHDR usage and syntax information instead of the control blocks.

ALL

Display function, operands, and syntax information for the IPHDR command. ALL is the default.

FUNCTION

Only display function information.

OPERANDS

Only display operand information.

SYNTAX

Only display syntax information.

Note: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them will be used.

Sample output

Following is sample output of the ICMPHDR command.

```
ICMPHDR 9D77428 256

      ICMPv6
      Type/Code      : ECHO Request      CheckSum: 4F51 0000
      Id              : 0028              Seq: 0
      Time            : 2002/05/23 18:43:00.332756
      Echo Data       : -8
000000 3CED3834 000513D4 08090A0B 0C0D0E0F |<.84.....|
000010 10111213 14151617 18191A1B 1C1D1E1F |.....|
000020 20212223 24252627 28292A2B 2C2D2E2F |!"#$%&'()*+,-./|
000030 30313233 34353637 38393A3B 3C3D3E3F |0123456789::<=>?|
000040 40414243 44454647 48494A4B 4C4D4E4F |@ABCDEFGHIJKLMNO|
000050 50515253 54555657 58595A5B 5C5D5E5F |PQRSTUVWXYZ.\.^_|
000060 60616263 64656667 68696A6B 6C6D6E6F |`abcdefghijklmno|
000070 70717273 74757677 78797A7B 7C7D7E7F |pqrstuvwxyz{|}~.|
000080 80818283 84858687 88898A8B 8C8D8E8F |.....|
000090 90919293 94959697 98999A9B 9C9D9E9F |.....|
0000A0 A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF |.....|
0000B0 B0B1B2B3 B4B5B6B7 B8B9BABB BCBDBEBF |.....|
0000C0 C0C1C2C3 C4C5C6C7 C8C9CACB CCCDCECF |.....|
0000D0 D0D1D2D3 D4D5D6D7 D8D9DADB DCDDDEDF |.....|
0000E0 E0E1E2E3 E4E5E6E7 E8E9EAEB ECEDEEEF |.....|
0000F0 F0F1F2F3 F4F5F6F7 F8F9FAFB FCFDFEFF |.....|

Protocol Header      : 8
000000 80004F51 00280000

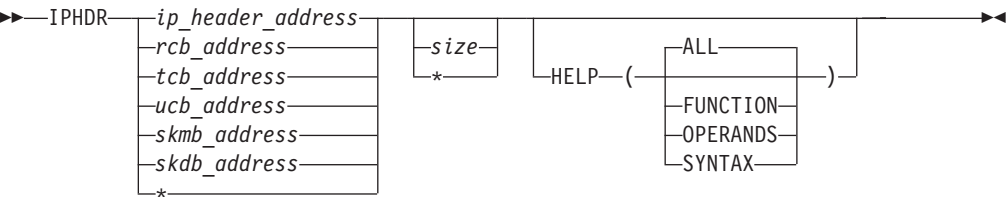
Data                  : 590      Data Length: 0
000000 3CED3834 000513D4 08090A0B 0C0D0E0F |.....M.....|
000010 10111213 14151617 18191A1B 1C1D1E1F |.....|
000020 20212223 24252627 28292A2B 2C2D2E2F |.....|
```

IPHDR

Invocation of the IPHDR command displays the IP header fields.

Syntax

Following is the syntax of the IPHDR subcommand:



Parameters

Following are the parameters of the IPHDR subcommand:

- * To omit this positional parameter when using the HELP keyword.

ip_header_address

The address of an IP header or the symbol for the address.

rcb_address

The address of a raw control block or the symbol for the address.

tcb_address

The address of a TCP/IP TCB control block or the symbol for the address.

ucb_address

The address of a UDP control block or the symbol for the address.

skmb_address

The address of an SKMB control block or the symbol for the address.

skdb_address

The address of an SKDB control block or the symbol for the address.

size

The amount of data to display. If the size is greater than the size of the header, additional protocol headers (if any) are displayed. Must be one to three hexadecimal digits.

HELP

Display IPHDR usage and syntax information instead of the control blocks.

ALL

Display function, operands, and syntax information for the IPHDR command.
ALL is the default.

FUNCTION

Only display function information.

OPERANDS

Only display operand information.

SYNTAX

Only display syntax information.

Note: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them will be used.

Sample output

The following is a sample output of the IPHDR command:

```
IPHDR 09D77400 300
```

```
IP Header: 09D77400
```

```
IpHeader: Version : 6           Header Length: 40
Class:           : 00           Flow: 000000
Payload Length   : 264
Hops              : 255          Protocol: ICMPv6
Source           : ::1
Destination      : ::1
```

```
ICMPv6
```

```
Type/Code       : ECHO Request   CheckSum: 4F51 0000
Id              : 0028           Seq: 0
Time            : 2002/05/23 18:43:00.332756
Echo Data       : 256
```

```

000000 3CED3834 000513D4 08090A0B 0C0D0E0F |<.84.....|
000010 10111213 14151617 18191A1B 1C1D1E1F |.....|
000020 20212223 24252627 28292A2B 2C2D2E2F |! "$%&'()*+,-./|
000030 30313233 34353637 38393A3B 3C3D3E3F |0123456789:;<=>?|
000040 40414243 44454647 48494A4B 4C4D4E4F |@ABCDEFGHIJKLMNO|
000050 50515253 54555657 58595A5B 5C5D5E5F |PQRSTUVWXYZ.\.^_|
000060 60616263 64656667 68696A6B 6C6D6E6F |`abcdefghijklmnopqrstuvwxyz{|}~.|
000070 70717273 74757677 78797A7B 7C7D7E7F |.....|
000080 80818283 84858687 88898A8B 8C8D8E8F |.....|
000090 90919293 94959697 98999A9B 9C9D9E9F |.....|
0000A0 A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF |.....|
0000B0 B0B1B2B3 B4B5B6B7 B8B9BABB BCBDBEBF |.....|
0000C0 C0C1C2C3 C4C5C6C7 C8C9CACB CCCDCECF |.....|
0000D0 D0D1D2D3 D4D5D6D7 D8D9DADB DCDDDEDF |.....|
0000E0 E0E1E2E3 E4E5E6E7 E8E9EAEB ECEDEEEF |.....|
0000F0 F0F1F2F3 F4F5F6F7 F8F9FAFB FCFDFEFF |.....|

IP Header          : 40
000000 60000000 01083AFF 00000000 00000000 00000000 00000001 00000000 00000000
000020 00000000 00000001

Protocol Header    : 8
000000 80004F51 00280000

Data              : 720    Data Length: 256
000000 3CED3834 000513D4 08090A0B 0C0D0E0F |.....M.....<.84.....|
000010 10111213 14151617 18191A1B 1C1D1E1F |.....|
000020 20212223 24252627 28292A2B 2C2D2E2F |.....! "$%&'()*+,-./|
000030 30313233 34353637 38393A3B 3C3D3E3F |.....0123456789:;<=>?|
000040 40414243 44454647 48494A4B 4C4D4E4F |.....$.<(+)@ABCDEFGHIJKLMNO|
000050 50515253 54555657 58595A5B 5C5D5E5F |&.....!$*);^PQRSTUVWXYZ.\.^_|
000060 60616263 64656667 68696A6B 6C6D6E6F |-/.....,%>?`abcdefghijklmnopqrstuvwxyz{|}~.|
000070 70717273 74757677 78797A7B 7C7D7E7F |.....~:#@`="pqrstuvwxyz{|}~.|
000080 80818283 84858687 88898A8B 8C8D8E8F |.abcdefghi.....|
000090 90919293 94959697 98999A9B 9C9D9E9F |.jklmnopqr.....|
0000A0 A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF |.stuvwxyz.....|
0000B0 B0B1B2B3 B4B5B6B7 B8B9BABB BCBDBEBF |.....|
0000C0 C0C1C2C3 C4C5C6C7 C8C9CACB CCCDCECF |.ABCDEFGHI.....|
0000D0 D0D1D2D3 D4D5D6D7 D8D9DADB DCDDDEDF |.JKLMNOPQR.....|
0000E0 E0E1E2E3 E4E5E6E7 E8E9EAEB ECEDEEEF |.STUVWXYZ.....|
0000F0 F0F1F2F3 F4F5F6F7 F8F9FAFB FCFDFEFF |0123456789.....|

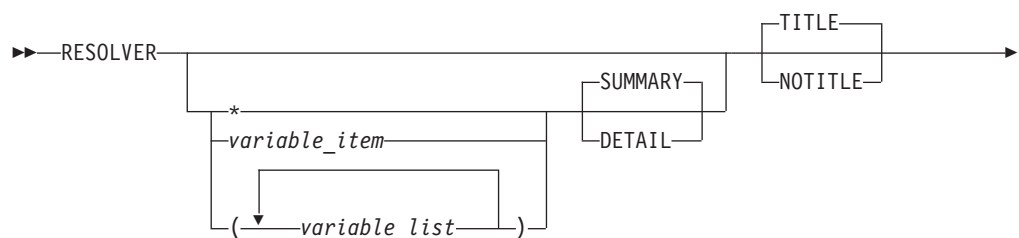
```

RESOLVER

Invocation of the RESOLVER command formats and summarizes resolver control blocks.

Syntax

Following is the syntax of the RESOLVER command:



DETAIL

In addition to the SUMMARY display, DETAIL also shows the contents of the control blocks.

TITLE

The title contains information about the dump and about the RESOLVER command. The title information is displayed as the default. The title contains the following information:

- RESOLVER command input parameters.
- Dump data set name.
- Dump title.

NOTITLE

Suppress the title lines. Use this when you are processing lots of commands on the same dump and do not need to see the title information repeated.

Notes:

1. If you specify multiple keywords from the set {TITLE,NOTITLE}, only the last one will be used.
2. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.
3. If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them will be used.

Sample output

The following is sample output of the RESOLVER command with only DETAIL specified:

```
| RESOLVER * DETAIL
| Dataset: IPCS.M999999.DUMPA
| Title:  RESOLVER V1R4:Job(BPXAS) EZBRERSR(HIP6140 20020503)+
|         000000D6 SOC1/00000001 P=0027,S=0027,H=0027
| =====
|
| Resolver Analysis
|
| RCRT: 0178B058
| +0000 RCRTENT#. 0000001F RCRTRCVT. 89E93000
|
| -- Array elements --
| +0008 RCRTENTS. 0178B128 RCRTENTS. 0178B1E0 RCRTENTS. 0178B298
| +0014 RCRTENTS. 0178B350 RCRTENTS. 0178B408 RCRTENTS. 0178B4C0
| +0020 RCRTENTS. 0178B578 RCRTENTS. 0178B630 RCRTENTS. 0178B6E8
| +002C RCRTENTS. 0178B7A0 RCRTENTS. 0178B858 RCRTENTS. 0178B910
| +0038 RCRTENTS. 0178B9C8 RCRTENTS. 0178BA16 RCRTENTS. 0178BA64
| +0044 RCRTENTS. 0178BB1C RCRTENTS. 0178BBD4 RCRTENTS. 0178BC8C
| +0050 RCRTENTS. 0178BD44 RCRTENTS. 0178BDFC RCRTENTS. 0178BEB4
| +005C RCRTENTS. 0178BF02 RCRTENTS. 0128FC00 RCRTENTS. 0128FC00
| +0068 RCRTENTS. 0128FC00 RCRTENTS. 0128FC00 RCRTENTS. 0128FC00
| +0074 RCRTENTS. 0128FC00 RCRTENTS. 0128FC00 RCRTENTS. 0178BF50
|
| -- End of array --
|
| RCVT: 89E93000
| +0000 RCVTID... RCVT      RCVTASCB. 00F89D00 RCVTETKN. 7FFCA358
| +000C RCVTLX... 00002B00 RCVTREFR. 00000004 RCVTTCA.. 7F6C3C68
| +0018 RCVTTOPT. FBFFFFFF RCVTRSMT. 093F4000 RCVTDEFA. 00000001
| +0024 RCVTDEFI. 00000002 RCVTGBLA. 00000003 RCVTGBLI. 00000004
|
| -- Array elements --
| +0030 RCVTCFG.. .....
| +006C .....
```

```

| +00A8 .....
| +00E4 .....
| +0120 .....
| +0130 RCVTCFG.. .....
| +016C .....
| +01A8 .....
| +01E4 .....
| +0220 .....
| +0230 RCVTCFG.. SYS1.TCPPARMS(TCPDATA).....
| +026C .....
| +02A8 .....
| +02E4 .....
| +0320 .....
| +0330 RCVTCFG.. .....
| +036C .....
| +03A8 .....
| +03E4 .....
| +0420 .....
| +0430 RCVTCFG.. .....
| +046C .....
| +04A8 .....
| +04E4 .....
| +0520 .....
| +0530 RCVTCFG.. .....
| +056C .....
| +05A8 .....
| +05E4 .....
| +0620 .....
| +0630 RCVTCFG.. .....
| +066C .....
| +06A8 .....
| +06E4 .....
| +0720 .....
| +0730 RCVTCFG.. .....
| +076C .....
| +07A8 .....
| +07E4 .....
| +0820 .....
|
|      -- End of array  --
|
| +0830          RCVTDEFD. 00000000  RCVTGBLD. 00000000
| +0838 RCVTDEFLHOSTD.          00000000
| +083C RCVTGBLLHOSTD.          00000000
| +0840 RCVTCART. 00000000 00000000  RCVTCNID. 00000001
| +084C RCVTINID. 00000000 RCVTABND. 00000000  RCVTJOB. RESOLVER
| +085C RCVTSTIM. B79291E8 7F3A95C8  RCVTPTIM. 00000000 00000000
| +0870 RCVTDEFLHOSTA.          00000005
| +0874 RCVTDEFLHOSTI.          00000006
| +0878 RCVTGBLLHOSTA.          00000007
| +087C RCVTGBLLHOSTI.          00000008
| +0880 RCVTFLAGSCS...          80000000
|
|
| Resolver Address Space is RESOLVER (ASID 01F8)
|
| Global TCPIP.DATA file: SYS1.TCPPARMS(TCPDATA)
|
| Default TCPIP.DATA file: None
|
| Global IPNODES file: None
|
| Default IPNODES file: None
|
|   CTCA: 7F6C3C68
| +0000 CTCA_CBID..... CTCA
| +0004 CTCA_CBSIZE.... 0398
| +0006 CTCA_CBVER..... 0001
| +0008 CTCA_CTNAME.... C5E9C2C3 C3E3D9C3 00F89D00 00000000

```



```

| +0018 CTCA_CTTOKEN.... B79291E8 7F6C3C68 0A448190 00F89D00
| +0028 CTCA_CURCTBS.... 7E674000
| +002C CTCA_CURRCD..... 7E67CE0A
| +0030 CTCA_TRACE..... 8A70CD98
| +0034 CTCA_OPTWORD.... 09E93018
| ...
| ...
| Resolver Task Data for USER557 Asid=0027 Tcb@=007F6AB0:
|
| RES_TASK: 7F6D6678
| +0000 RES_IDENTIFIER..... RTSK
| +0004 RES_LENGTH..... 0BB8
| +0006 RES_SUBPOOL..... F9
| +0007 RES_USERKEY..... 06
| +0008 RES_SEQUENCE#..... 00000004
| +0010 RESMGR_TOKEN..... 00000000
| +0014 RESMGR_DATA..... 00000000
| +0018 RES_RTsk@..... 7F6D6678
| +001C RES_RPID@..... 7F6D7230
| RES_STATE.....
| +0030 0000000A 00000001 000002C3 00000001 10020035 09438052 00000000
| +004C 00000000 00000000 00000000 00000000 00000000 00000000 00000000
| +0068 00000000 00000000 00000000 00000000 00000000 00000000 00000000
| +0084 00000000 00000000 00000000 00000000 00000000 00000000 00000000
| +00A0 00000000 00000000 00000000 00000000 00000000 00000000 00000000
| ...
| ...
| +02D0 00000000 7F6D6950
| RES_STATE_EXT.....
| +02D8 00000000 7F6D6BF8 00000000 7F6D6AF8 D4E5E2D9 00404040 00000000
| +02F4 00000000 00000000 00000000 00000000 00000000 00000000 00000000
| +0310 00000000 00000000 00000000 00000000 00000000 00000000 E3C3D7C3
| +032C E2000000 00E4E2C5 D9F5F500 00000000 00000000 00000000 00000000
| +0348 00000000 00000000 00000000 00000000 00000000 00000000 00000000
| +0364 00000000 00000000 00000000 00000000 00000000 00000000 00000000
| ...
| ...
| +0680 RES_SEARCH_COUNT.... 00000003
|
| -- Array elements --
| RES_SEARCH_ENTRY....
| +0684 RALEIGH.IBM.COM.....
| +06CA .....
| +0710 .....
| +0756 .....
| RES_SEARCH_ENTRY....
| +0784 IBM.COM.....
| +07CA .....
| +0810 .....
| +0856 .....
| RES_SEARCH_ENTRY....
| ...
| ...
| RRST: 7F6D66A8
| +0000 RETRANS.. 0000000A RETRY.... 00000001 OPTIONS.. 000002C3
| +000C NSCOUNT.. 00000001
|
| -- Array elements --
| +0010 NSADDR_LIST... 10020035 09438052 00000000 00000000
| +0020 NSADDR_LIST... 00000000 00000000 00000000 00000000
| +0030 NSADDR_LIST... 00000000 00000000 00000000 00000000
| +0040 NSADDR_LIST... 00000000 00000000 00000000 00000000
| +0050 NSADDR_LIST... 00000000 00000000 00000000 00000000
| +0060 NSADDR_LIST... 00000000 00000000 00000000 00000000
| +0070 NSADDR_LIST... 00000000 00000000 00000000 00000000
| +0080 NSADDR_LIST... 00000000 00000000 00000000 00000000
| +0090 NSADDR_LIST... 00000000 00000000 00000000 00000000

```

```

| +00A0 NSADDR_LIST...          00000000 00000000 00000000 00000000
| ...
| ...
| +02A0 RES_VERSION...          00000000
| +02A4 RES_EXTENSION.          7F6D6950
|
| RRSX: 7F6D6950
| +0004 STAT_EBCDICTOASCII..... 7F6D6BF8
| +000C STAT_ASCIIITOEBDIC..... 7F6D6AF8
| STAT_HOSTNAME.....
| +0010 MVS. ....
| +0050 STAT_SERVICENAME..... TCPS....
| +0059 STAT_C_DSPRF..... USER55.....
| +0078 $__IPDBCSNUM..... 00000000
| ...

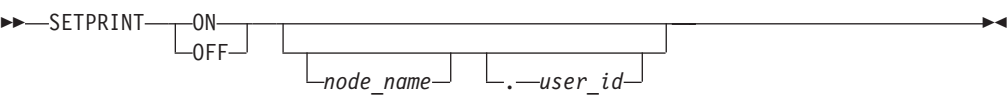
```

SETPRINT

Invocation of the SETPRINT command can change the destination of subsequent IPCS command output. If the IPCSPRNT data set is allocated and being sent to a node, the output of future IPCS commands will be accumulated (not displayed at the terminal) until you exit IPCS. When you exit IPCS, the IPCSPRNT data set is sent to the specified node.

Syntax

Following is the syntax of the SETPRINT command:



Parameters

Following are the parameters of the SETPRINT command:

ON

Allocates the IPCSPRNT data set and issues the IPCS command SETDEF PRINT.

OFF

Frees the IPCSPRNT data set and issues the IPCS command SETDEF NOPRINT.

node_name

The name of a TSO or VM system to which the output is sent.

user_id

The user ID on the TSO or VM system to which the output is sent.

Note: If *user_id* is specified, there must be a period but no space between *node_name* and *user_id*.

Sample output

If the command completes successfully, there is no output for the SETPRINT command. The following examples are invalid invocations of the SETPRINT command:

- **Example 1:** Allocating IPCSPRNT when it is already allocated.

```

setprint on ralvms.testid
IKJ56861I  FILE IPCSPRNT NOT UNALLOCATED, DATA SET IS OPEN

```

- **Example 2:** Freeing IPCSPRINT when it is already freed.

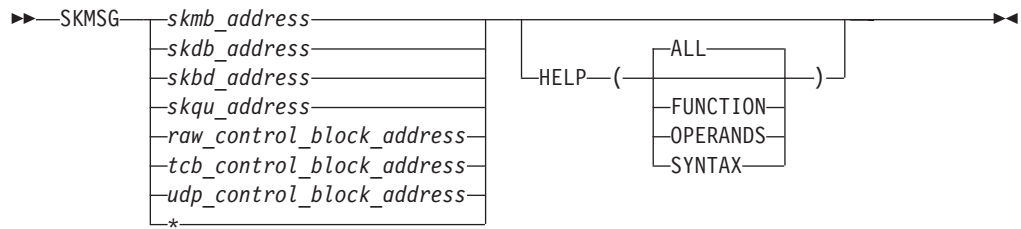
```
setprint off
BLS21060I PRINT file not open
IKJ56247I FILE IPCSPRINT NOT FREED, IS NOT ALLOCATED
```

SKMSG

Invocation of the SKMSG command displays the SKMSG fields.

Syntax

Following is the syntax of the SKMSG command:



Parameters

Following are the parameters of the SKMSG command:

- * To omit this positional parameter when using the HELP keyword.

skmb_address

The address of an SKMB control block or the symbol for the address.

skdb_address

The address of an SKDB control block or the symbol for the address.

skbd_address

The address of an SKBD control block or the symbol for the address.

skqu_address

The address of an SKQU control block or the symbol for the address.

raw_control_block_address

The address of a RAW control block or the symbol for the address.

tcb_control_block_address

The address of a TCB control block or the symbol for the address.

udp_control_block_address

The address of a UDP control block or the symbol for the address.

HELP

Display SKMSG usage and syntax information.

ALL

Displays help about the function, operands, and syntax information for the SKMSG command. ALL is the default.

FUNCTION

Only displays function help information.

OPERANDS

Only displays operands help information.

SYNTAX

Only displays syntax help information.

Note: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them will be used.

Sample Output

The following is a sample output of the SKMSG command:

SKMSG 15D4D5B8

SKDB at 15D4D5B8

Message 1

```
SKMB: 15D4D588
+0000 id..... SKMB      next..... 00000000 prev..... 00000000
+0008 tail..... 00000000 cont..... 15D55B08 flag..... 4000
+0012 band..... 00       strx..... 16      hold..... 00000000
+0018 datb..... 15D4D5B8 atch..... 00000000 rptr..... 15DE5A60
+0024 wptr..... 15DE5AA8
```

```
SKDB: 15D4D5B8
+0000 id..... SKDB      msgb..... 15D4D588 cver..... 00
+0009 csrc..... 80      ctyp..... 40
+000C tokn..... 15E3F040 15E3F3E0 00001358 alet..... 00000000
+001C base..... 15DE5000 size..... 00001000 flag..... 00000000
+0028 ref..... 00000005
```

```
Buffer: 15DE5000
+0000 450005DC 24760000 4006DBF0 09433116 | ..... ..0....
+0010 0943311A 0AB70866 481080F3 450C8271 | .....3..b.
+0020 8010FFFE DA3B0000 0101080A 3E456F23 | .....?..
+0030 3E450C82 91A38897 D3C7E5D6 C297E8E3 | ...bjthpLGVOBpYT
+0040 D9F0E596 F2C989D9 | R0Vo2IiR
```

```
SKMB: 15D55B08
+0000 id..... SKMB      next..... 00000000 prev..... 00000000
+0008 tail..... 00000000 cont..... 00000000 flag..... 0000
+0012 band..... 00       strx..... 00      hold..... 00000000
+0018 datb..... 15D55B38 atch..... 00000000 rptr..... 13ECA758
+0024 wptr..... 13ECACB8
```

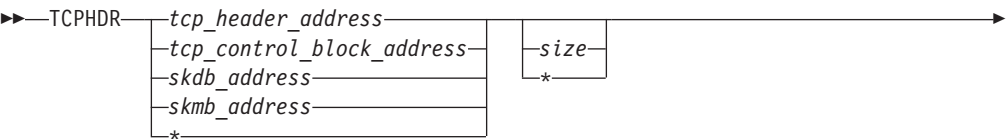
```
SKDB: 15D55B38
+0000 id..... SKDB      msgb..... 15D55B08 cver..... 00
+0009 csrc..... 40      ctyp..... 80
+000C tokn..... 15D41040 15D417F0 000003C7 alet..... 01FF0007
+001C base..... 13ECA000 size..... 00000CB8 flag..... 00800000
+0028 ref..... 00000003
```

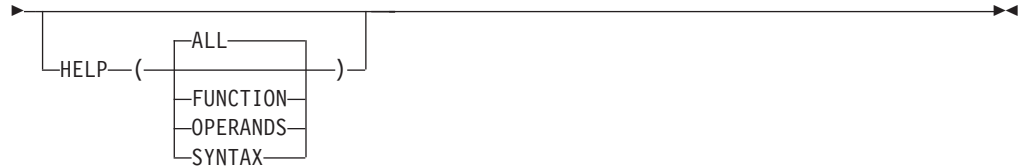
TCPHDR

Invocation of the TCPHDR command displays the TCP header fields.

Syntax

Following is the syntax of the TCPHDR command:





Parameters

Following are the parameters of the TCPHDR subcommand:

- * To omit this positional parameter when using the HELP keyword.

tcp_header_address

The address of the TCP header or an IPCS symbol.

tcp_control_block_address

The address of a TCP/IP TCP control block or an IPCS symbol.

skdb_address

The address of an SKDB control block or an IPCS symbol.

skmb_address

The address of an SKMB control block or an IPCS symbol.

size

The amount of data to display. If the size is greater than the size of the header, the variable portion of the header (if it exists) is displayed. Must be one to three hexadecimal digits.

HELP

Display TCPHDR usage and syntax information.

ALL

Displays help about the function, operands, and syntax information for the TCPHDR command. ALL is the default.

FUNCTION

Only displays function help information.

OPERANDS

Only displays operands help information.

SYNTAX

Only displays syntax help information.

Note: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them will be used.

Sample output

The following is sample output of the TCPHDR command:

```
TCPHDR 7F522108
```

```
TCB at 7F522108
```

```
TCP Header at 7F5222D8
```

```
7F5222D8 04010402 7228DD16 7228DB82 50107FD8 | .....b&."Q |
+0010 00000000 | ....
```

```
Source Port      : 1025
Destination Port : 1026
Sequence Number  : 1,915,280,662
```

```

Ack Number      : 1,915,280,258
Header Length   : 20
Flags           : Ack
Window Size     : 32728
Checksum        : 0000
Urgent Data Pointer : 0000

```

TOD

Invocation of the TOD command formats a hexadecimal time-of-day value into a readable date and time.

Syntax

Following is the syntax of the TOD command:

```

>>—TOD—time_value—┐
                      └,—time_zone—┘

```

Parameters

Following are the parameters of the TOD subcommand:

time_value

The time to be converted. *The time_value* can be specified as either 16 hexadecimal digits or as an address in a dump of an eight-byte STCK value. If less than 16 digits are specified, the value is padded on the right with zeros. If an address is specified, it must be followed by a period. If an address is less than eight hexadecimal digits, it is padded on the left with zeros.

time_zone

An offset for the time (the difference between local time and GMT). The *time_zone* can be specified either as a word or as a positive or negative decimal value. The recognized words are:

LOCAL

Time zone value of zero is used. This is the default.

GMT Greenwich Mean Time

EDT U.S. Eastern Daylight Time zone

EST U.S. Eastern Standard Time zone

CDT U.S. Central Daylight Time zone

CST U.S. Central Standard Time zone

MDT U.S. Mountain Daylight Time zone

MST U.S. Mountain Standard Time zone

PDT U.S. Pacific Daylight Time zone

PST U.S. Pacific Standard Time zone

Sample output

The following are sample outputs of the TOD command:

- **Example 1:** STCK time-of-day with a time zone word.

```
Command ==> ip tod b214030791f3a92c,est
```

```
B2140307 91F3A92C : 1999/04/10 20:51:58.684986  TIMEZONE: 0000430E23400000
```

- **Example 2:** An address in the dump where an STCK time-of-day value is located with a negative time zone offset.

Command ==> ip tod 11275d4.,-4

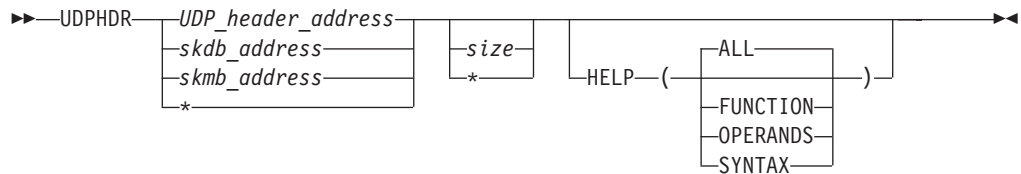
B24000E0 51900000 : 1999/05/16 05:36:37.632256 TIMEZONE: FFFCA5B17000000

UDPHDR

Invocation of the UDPHDR command displays the UDP header fields.

Syntax

Following is the syntax of the UDPHDR command:



Parameters

Following are the parameters of the UDPHDR subcommand:

- * To omit this positional parameter when using the HELP keyword.

UDP_header_address

The address of a UDP header or the symbol for the address.

Note: The UDP header has no version or identifier, so it is not possible to definitively recognize a UDP header given an address in storage. Therefore, this command will format the storage assuming it is a UDP header.

skdb_address

The address of an SKDB control block or the symbol for the address.

skmb_address

The address of an SKMB control block or the symbol for the address.

HELP

Display UDPHDR usage and syntax information.

ALL

Displays help about the function, operands, and syntax information for the UDPHDR command. ALL is the default.

FUNCTION

Only displays function help information.

OPERANDS

Only displays operands help information.

SYNTAX

Only displays syntax help information.

Note: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them will be used.

Sample output

The following is a sample output of the UDPHDR command:

UDPHDR 08D0A0D8

UDP Header at 08D0A0EC

08D0A0EC 040700A1 0033CD23 | ...~.... |

Source port : 1031
Destination port : 161
Datagram Length : 51
Checksum : CD23

Installing TCP/IP IPCS subcommands

Installation has two parts:

1. Install the members of the target data sets.
2. Optionally, connect the TCP/IP panels to your existing ISPF panels.

Table 14 shows the target data sets that contain the data necessary to set up the TCP/IP IPCS subcommands. You need to copy the members to another data set or concatenate the target data sets into the DDNAME statements shown. Note that the target data sets contain other members, so you may not want to simply concatenate the target data set. A simple installation method is to change the TSO logon procedure to concatenate these data sets.

Table 14. Target data sets for TCP/IP IPCS subcommands

Target data set	Members	DD Name	Description
hlq.HELP	EZBIPCSH EZBIIHDH EZBIMHDH EZBIRESH EZBISKMH EZBITHDH EZBIUHDH SYSTCPDA SYSTCPIS	SYSHELP	TCPIPICS HELP text IPHDR Help Text ICMPHDR Help Text RESOLVER Help Text SKMSG Help Text TCPHDR Help Text UDPHDR Help Text SYSTCPDA Help Text SYSTCPIS Help Text
hlq.PARMLIB	EZAIPCSP	PARMLIB	IPCS verbexit mappings

Table 14. Target data sets for TCP/IP IPCS subcommands (continued)

Target data set	Members	DD Name	Description
<i>hlq</i> .SBLSCLI0	EZATFTHD (alias TCPHDR) EZATSPRI (alias SETPRINT) EZBTCPEX EZBTDENO (alias ERRNO) EZBTFICH (alias ICMPHDR) EZBTFIPH (alias IPHDR) EZBTFSKM (alias SKMSG) EZBTFTOD (alias TOD) EZBTFUPH (alias UDPHDR) EZBTIPCS (alias TCIPCS) EZBTRESO (alias RESOLVER)	SYSEXEC	REXX execs
<i>hlq</i> .MIGLIB	EZBDGIPC	STEPLIB or IPCS TASKLIB	Load module
<i>hlq</i> .SBLSPNL0	EZBD* (approximately 170 members)	ISPPLIB	ISPF panels
<i>hlq</i> .SBLSTBL0	EZBDKEYS	ISPTLIB	ISPF key lists
<i>hlq</i> .SBLMSG0	EZBF* (6 members)	ISPLIB	ISPF messages

To use the panel interface to the TCP/IP IPCS subcommands, you can either invoke the panels using an IPCS command or connect the TCP/IP ISPF panels to an existing ISPF panel. No additional installation steps are required to invoke the panels using an IPCS command. To connect the TCP/IP ISPF panels to an existing panel, find an existing panel where you wish to add TCP/IP as an option and modify the panel. Modify the panel by adding the TCP/IP option, which will invoke the following command:

```
PGM(BLSGSCMD) PARM(%EZBTCPEX) NEWAPPL(EZBD)
```

where BLSGSCMD is the IPCS command, EZBTCPEX is the TCP/IP REXX exec, and EZBD is the TCP/IP key list prefix.

Entering TCP/IP IPCS subcommands

The TCP/IP IPCS subcommands can be entered as an IPCS command, using panels provided by TCP/IP, or by using the IPCS batch facility.

Follow these steps to enter a TCP/IP IPCS subcommand (you can use the IPCS Subcommand Entry panel).

1. Log on to TSO.
2. Access IPCS to display the IPCS Primary Option Menu. Figure 15 shows an example of an IPCS Primary Option Menu.

```

----- IPCS PRIMARY OPTION MENU -----
OPTION  ==>
 0  DEFAULTS  - Specify default dump and options
 1  BROWSE    - Browse dump data set
 2  ANALYSIS  - Analyze dump contents
 3  UTILITY   - Perform utility functions
 4  COMMAND   - Enter IPCS subcommand or CLIST
 5  TCP/IP    - TCP/IP analysis commands
 6  NCP       - NCP analysis commands
 7  NMP       - NMP analysis commands
 8  INVENTORY - Inventory of problem data
 9  SUBMIT    - Submit problem analysis job to batch
 T  TUTORIAL  - Learn how to use the IPCS dialog
 X  EXIT      - Terminate using log and list defaults

Enter END command to terminate IPCS dialog

```

Figure 15. IPCS primary option menu

3. Select option 4, COMMAND.
4. Type the TCP/IP IPCS subcommand. Figure 16 shows the IPCS Subcommand Entry panel with a subcommand entered.

```

----- IPCS Subcommand Entry -----
Enter a free-form IPCS subcommand or a CLIST or REXX exec invocation below:

==> tcpipcs help

----- IPCS Subcommands and Abbreviations -----
ADDUMP, DROP D  LISTMAP, LMAP  RUNCHAIN, RUN C
ANALYZE, DROP M  LISTSYM, LSYM  SCAN
ARCHEXIT, DROP S  LISTUCB, LIST U  SELECT
ASCBEXIT, EQU, EQ  LITERAL  SETDEF, SET D
ASMCHK, F  LPAMAP  STACK
CBFORMAT, CB  MERGE  STATUS, ST
CBSTAT, FINDMOD, FMOD  NAME  SUMMARY, SUMM
CLOSE, FINDUCB, FIND U  NAMETOKN  SYSTRACE
COPYDDIR, INTEGER  NOTE, N  TCBEXIT, TCBX
COPYDUMP, IPCH HELP, H  OPEN  WEBEXIT, WEBBX
COPYTRC, LIST, L  PROFILE, PROF  WHERE, W
CTRACE, LISTDUMP, LDMP  RENUM, REN

```

Figure 16. IPCS subcommand entry panel with a TCP/IP IPCS subcommand entered

There are two ways to invoke the TCP/IP IPCS panels.

- One way is to invoke the panel REXX exec as an IPCS Subcommand. Follow the steps above for entering a TCP/IP IPCS subcommand using the IPCS Subcommand Entry panel and enter the command:
EZBTCPEX
- The second way to invoke the TCP/IP IPCS panels is to select the option provided in the installation section above.

Either way you invoke the panels, you should see the main menu for the TCP/IP IPCS commands shown in Figure 17 on page 281. Select an option and the panels will prompt you for additional menu choices or input for the specific TCP/IP IPCS

subcommand you select. After all input has been selected, the TCP/IP IPCS subcommand will be invoked using the current default dump data set.

```

                                TCP/IP Analysis Menu
Command ==>
(C) Copyright IBM Corporation 1998,2001. All rights reserved.
Select one of the following. Then press Enter.

1. General . . . - HEADER, MTABLE, STATE, TSDB, TSDX, TSEB
2. Protocol . . - PROTOCOL, RAW, TCB, UDP
3. Configuration - CONFIG, CONNECTION, PROFILE, ROUTE
4. Resources . . - COUNTERS, LOCK, MAP, STORAGE, TIMER
5. Execution . . - DUAF, DUCB, TRACE
6. Interfaces . - API, SOCKET, STREAM
7. Structures . - HASH, TREE
8. Functions . . - FIREWALL, FRCA, POLICY, TELNET, VMCF, XCF
9. Headers . . . - ICMPHDR, IPHDR, SKMSG, TCPHDR, UDPHDR
10. Converters . - ERRNO, SETPRINT, TOD
11. Applications. - RESOLVER
```

Figure 17. Main menu for TCP/IP IPCS subcommands.

Refer to *z/OS Communications Server: SNA Migration* for information regarding VTAM IPCS commands.

Using the batch option

Perform this step to access IPCS commands using the batch processing interface.

Step	Action
1	Prepare the JCL data set. Refer to the <i>z/OS MVS IPCS User's Guide</i> and <i>z/OS MVS IPCS Commands</i> .
Example	Sample command (single command): %TCPIPCS TELNET (* DETAIL)

Part 3. Diagnosing z/OS CS components

Chapter 7. Diagnosing line print requester and daemon (LPR and LPD) problems

Line print requester (LPR) and line printer daemon (LPD) compose a functional unit in which the LPR client sends data files to a printer controlled by an LPD server. These files can be in ASCII form or extended binary-coded decimal interchange code (EBCDIC) form.

In most environments, customers have different types of LPR clients and LPD servers, running on platforms, such as MVS, OS/2, AIX, and UNIX. However, all print client and servers must follow the standards contained in RFC1179. Some clients and servers provide more than what is required by the RFC, while some clients and servers are restricted or limited, which may cause errors or require more configuration to work.

On platforms, such as MVS, UNIX, and AIX, you can start the LPR client program with command prompts, through batch (in MVS), or through shell scripts (in UNIX/AIX®). The MVS LPD server allocates temporary data sets to process incoming print requests from various clients. These data sets use the TCP/IP high level qualifiers (HLQs) or the prefix defined in the LPD server cataloged procedure.

The MVS LPD server can also act as a client when a remote print server is defined in the LPD configuration file as a *service*. In this case, when the LPD server receives an incoming print job, it opens a new connection through a client port, and sends the data to the remote print server. When a remote print server is used, LPD specifications, such as line size and page size, do not apply. Instead, the specifications of the remote server apply.

For information on configuring your LPD server, refer to the *z/OS Communications Server: IP Configuration Reference*. For information on using the client-related LPR, LPQ, and LPRM commands, refer to the *z/OS Communications Server: IP User's Guide and Commands*.

Problems with the print function are usually easy to diagnosis if the problem is within the LPR client or the LPD server. More difficult problems may be encountered in the TCP/IP layer or in sockets. In addition, incorrectly built or defined translation tables can produce unpredictable results, such as abends, misprinted characters, and hang conditions (usually caused by delayed acknowledgments).

Diagnosing LPR client and LPD server problems

Problems with LPR and LPD generally fall into one of the following categories:

- Abends
- Timeouts, hangs, and waits
- Incorrect output

These categories are described in the following sections.

Abends

When an abend occurs during LPD processing, messages and other error-related information are sent to the MVS system console. If this information is insufficient to solve the problem, use the information provided in a dump. To produce a dump, code a SYSMDUMP DD or SYSABEND DD statement in the LPD cataloged

procedure. If you do not do the coding before the abend occurs, code the statement after the abend, recreate the abend or wait for it to occur again. For information about analyzing dumps produced during LPD processing, refer to *z/OS MVS Diagnosis: Procedures*.

It can also be helpful to obtain and analyze information from the following sources:

- LPD trace in the SYSPRINT data set
- Output of LPD started task
- System log (syslog)

Timeouts, hangs, and waits

Timeouts, hangs, and waits occur when the LPD server does not respond to client requests for a data packet, an acknowledgment that a data packet was received, or a reply to a command. Similarly, the LPD server can time out a connection if the LPR client does not respond.

One or more of the following problems can cause a timeout, hang, or wait:

- Incorrect host name or IP address specified on the LPR command
- Malfunctioning remote server or remote host
- Problems with the network (for example, network congestion), bridge, gateway, or router in the routing path
- Problems with the device or channel attached to the host
- Corrupted TCP/IP address space
- Incorrectly built or defined translation tables
- Malfunctioning LPR client

If a timeout, hang, or wait occurs, do one or more of the following.

1. Check to see if the target LPD print server is running, has enough paper, and is not jammed.
2. Check the LPR and LPD traces for possible error messages, or for the last activity performed by LPR or LPD (for example, waiting for a connection, port availability, or an acknowledgment). Be aware that when sending a print request to a remote printer through the LPD server, the LPR client can show a successful data transfer even though there may be a problem connecting to the remote printer.
3. Check the IP address or host name used with the LPR command.
4. Check the LPR, LPD, and packet traces. If the packet trace shows a problem during binding or connecting, then check the socket trace.
5. Verify that the translation tables are built correctly. Test them using the *hlq.STANDARD.TCPXLBIN* table supplied with TCP/IP.

Be aware that waits can occur because some LPD servers do not send acknowledgments until data is actually printed. In this situation, the LPR client does not show successful data transfer until it actually receives the acknowledgment.

Incorrect output

LPR problems with incorrect output usually fall into one of the following categories:

- Garbled data sent from the LPR client or received by the LPD server
- Truncated or missing print data
- LPR works with some options, but not others

These categories are described in the following sections.

Garbled data

If garbled data is the problem, do one or more of the following:

1. Determine whether the binary option or the default EBCDIC was used when the data file was printed. If the binary option was used, the LPR client did not translate the data. If EBCDIC was used, check for erroneous control characters or conflicting combinations of options.
2. Check to see if other files print correctly from the same client and to the same server. Check to see if the problem file prints correctly to other servers.
3. Verify that the translate tables for the sender and receiver are reciprocals of each other. Determine which characters are consistently garbled and examine those entries in the tables. To determine the name of the translation table used by the LPR client, check the LPR messages issued at startup.
4. Check the IP packet trace to determine exactly what data was sent from the client and acknowledged by the LPD server.
5. If data shown in the IP packets from the LPR client to the server is correct, there may be an error on the server or printer. Check the server traces and setup on the printer or LPD server. Some servers require certain printer names or options to be specified on the LPR (lp from omvs) commands.

Truncated or missing print data

If truncated or missing print data is the problem, do one of the following:

1. Check to see if the value for the record length is valid. The value is specified using the WIDTH option and variable on the LPR command.
2. If MVS displays truncated records, check the value of the LINESIZE option on the SERVICE statement in the LPD configuration file.
3. If you use the FILTER L or FILTER R options on the LPR command, check to see if the control characters on the first column of the source file are valid. LPR issues a message indicating whether a record of data has been ignored.
4. Using a packet trace and the file size listed in the LPR trace control record, verify that the correct number of bytes were sent by the LPR client and received by the LPD server.
5. Check the LPD trace for error messages. Verify that the Job xxx Received and Job xxx Finish Printing messages were received.
6. If sending a print request to a remote printer through the LPD server, check the LPD trace to determine if all data were sent successfully to the remote printer. If not or if data are incorrect, check the printer for errors or restrictions on the type of data it supports (for example, postscript only, text only, and so on).
7. Check for partial temporary data sets and either rename them or delete them. The LPD server creates temporary data sets when connections are broken, and the server does not completely process a print job. (Depending on the LPR client, the server can requeue the job for printing at a later time.) When the connection is restored, the daemon checks for temporary data sets and processes them. After processing, they are erased.

The temporary data sets are stored on a volume with a data set prefix you define in the LPD cataloged procedure. Following are samples of these data sets:

TCPUSR4.PRT1.QUEUE		WRKLB2
TCPUSR4.RALVM12.CFn	BROWSED	WRKLB2
TCPUSR4.RALVM12.DFA	BROWSED	TCPWRK
TCPUSR4.RALVM12.JFn		WRKLB2

The QUEUE... represents, in this sample PRT1's print queue file. It will contain the name of

the JOB files that have not been completely processed yet.
 The CF... represents the CONTROL FILE.
 Contains the control data/commands sent to LPD.
 The DF... represents the DATA FILE.
 The actual data sent to be printed.
 The JF... represents the JOB FILE.
 Contains names of the above files that have not been processed yet.
 where nnn is the three digit job number.

Occasionally, depending on the precipitating incident and the time the connection was broken, the LPD server creates only a portion of one or more data sets. When partial temporary data sets are created, the server issues allocation or failure-to-erase messages. If you receive any of these messages, search for the partial data sets and either rename or delete them. After doing this, you may need to reissue the print request or requests.

The LPD trace and the system log at the time a connection is broken show the status of all print jobs (and the status of some data sets) and identify the owners of the print requests.

LPR works with some options only

If the LPR command works with some options, but not with others, do one or more of the following:

- If some print requests do not work with certain LPR options, check the LPR trace for error messages.
- If the LPR command from batch fails, but works under TSO, check for possible errors in the batch-job output and for error messages in the LPR trace.

For information about the LPR command, refer to the *z/OS Communications Server: IP User's Guide and Commands*.

LPR client traces

This section provides information about activating LPR client traces. It also provides samples of trace output with explanations of selected messages.

Activating LPR client traces

You can activate LPR client traces by specifying the TRACE option in addition to the usual processing parameters on the LPR command.

For example, to start the LPR client with trace on, enter the following command:

```
LPR filename (Printer p1 Host h1 TRACE
```

Client trace output

LPR trace output is sent to SYSOUT and can be displayed on the LPR client console. Figure 18 on page 289 is a sample of an LPR trace created by way of TSO with the following command:

```
LPR soto.files(lpconfig) (p prt1 h 9.67.113.60 TRACE
```

```

EZB0915I Begin "LPR" to printer "prt1" at host "9.67.113.60"
1
EZB1057I Loaded translation table from "TCP31S.STANDARD.TC PXLBIN".
EZB0920I Requesting TCP/IP service at 96155 18:52:53
EZB0921I Granted TCP/IP service at 96155 18:52:53
EZB0922I Resolving 9.67.113.60 at 96155 18:52:53
EZB0924I Host 9.67.113.60 name resolved to 9.67.113.60 at 96155 18:52:53
EZB0925I TCP/IP turned on.
EZB0926I Host "MVSA" Domain "TCP.RALEIGH.IBM.COM" TCPIP Service Machine TCP31S
EZB0927I Trying to open with local port 721 to foreign host address 9.67.113.60
2
EZB0928I Connection open from local port 721 to foreign host address 9.67.113.60
EZB0961I Control file name is cfa827MVSA
EZB0962I Data file name is dfa827MVSA Port Number=721. Remote IP Addr=9.7.113.60
3
EZB0916I Sending command 2 argument: prt1 Port Number=721. Remote IP Addr=9.67.113.60
EZB0917I Command successfully sent Port Number=721. Remote IP Addr=9.67.113.60
EZB1012I Receiving ACK Port Number=721. Remote IP Addr=9.67.113.60
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
EZB0997I Byte size check starts at 96155 18:52:54
EZB0998I Byte size check ends at 96155 18:52:54
EZB0999I Send command starts at 96155 18:52:54 Port Number=721. Remote IP Addr=9.67.113.60
4
EZB0916I Sending command 3 argument:7434 dfa827MVSA Port Number=721. Remote IP Addr=9.67.113.60
EZB0917I Command successfully sent Port Number=721. Remote IP Addr=9.67.113.60
5
EZB1012I Receiving ACK Port Number=721. Remote IP Addr=9.67.113.60
5
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
EZB1000I Send command ends at 96155 18:52:55 Port Number=721. Remote IP Addr=9.67.113.60
6
EZB1001I Send data starts at 96155 18:52:55 Port Number=721. Remote IP Addr=9.67.113.60
6
EZB1002I Send data ends at 96155 18:52:56 Port Number=721. Remote IP Addr=9.67.113.60
EZB1003I Send ACK starts at 96155 18:52:56 Port Number=721. Remote IP Addr=9.67.113.60
EZB1014I Sending ACK Port Number=721. Remote IP Addr=9.67.113.60
7
EZB1015I ACK successfully sent Port Number=721. Remote IP Addr=9.67.113.60
EZB1004I Send ACK ends at 96155 18:52:56 Port Number=721. Remote IP Addr=9.67.113.60
EZB1012I Receiving ACK Port Number=721. Remote IP Addr=9.67.113.60
8
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
9
EZB1009I Data file sent. Port Number=721. Remote IP Addr=9.67.113.60
EZB1011I Queuing control line "HMSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "PTCPUSR4"
EZB1011I Queuing control line "JTCPUSR4.SOTO.FILES(LPCONFIG)"
EZB1011I Queuing control line "CMVSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "LTCPUSR4"

```

Figure 18. Example of LPR trace output (Part 1 of 2)

```

10
EZB1011I Queuing control line "fdfA827MVSA"
EZB1011I Queuing control line "UdfA827MVSA"
EZB1011I Queuing control line "NTCPUSR4.SOTO.FILES/LPCONFIG"
11
EZB0916I Sending command 2 argument: 153 cfA827MVSA Port Number=721. Remote IP Addr=9.67.113.60
EZB0917I Command successfully sent Port Number=721. Remote IP Addr =9.67.113.60
EZB1012I Receiving ACK Port Number=721. Remote IP Addr=9.6 7.113.60
12
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
13
EZB1017I Control data sent Port Number=721. Remote IP Addr=9.67.113.60
EZB1014I Sending ACK Port Number=721. Remote IP Addr=9.67. 113.60
EZB1015I ACK successfully sent Port Number=721. Remote IP Addr=9.67.113.60
EZB1012I Receiving ACK Port Number=721. Remote IP Addr=9.6 7.113.60
14
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
15
EZB1018I Control file sent Port Number=721. Remote IP Addr=9.67.113.60

```

Figure 18. Example of LPR trace output (Part 2 of 2)

Following are short descriptions of the numbered items in the trace:

- 1** Indicates the translation table used by the LPR client. In this print request, no translation tables were defined by the person submitting the request.
- 2** Indicates LPR port used to connect to the LPD server with the IP address 9.67.113.60. The LPR port range is from 721 through 731.
- 3** Indicates the LPR command sent to the LPD server identifying the name of the print queue where the output was sent. Refer to RFC1179 for details on commands and subcommands issued between LPR and LPD.
- 4** Indicates the command that provided the LPD print server with the byte size (7434) and name of the data file (dfA827MVSA) that was sent.
 - The character string dfA indicates that this was a data file.
 - The number 827 was the three-digit job number that was randomly generated by the LPR client or specified in the LPR command using the **JNUM** option.
 - MVSA was the name of the host from which the print request came.
- 5** Indicates the client is waiting for the LPD server to acknowledge the sending command in item **4**. The message on the following line (TRUE (00)) indicates that the client received an acknowledgment. A FALSE message or any value other than zero terminates the LPR print request.
- 6** Indicates that the LPR client started and then stopped sending the data file.
- 7** Indicates that the LPR client notified the LPD server, by way of an acknowledgment, that the complete file was sent. The LPR client waits for the server to acknowledge receipt of the entire data file.
- 8** Indicates that the client received an acknowledgment from the server that the entire data field was received.
- 9** Confirms that the data file was sent to the LPD server.
- 10** Specifies one of the several control records sent by the LPR client. (The records are described in detail in RFC1179.) This control record is mandatory and represents the name of the data file created by the LPD server. The name is preceded by the filter specified on the LPR command. The letter **f** denotes the default filter.

- 11** Specifies the byte size (153) and the name of the control file (cfA827MVSA) that was sent.
- 12** Indicates that the LPD server received the command and expected the control file to be sent.
- 13** Indicates that the LPR client sent the control file and an acknowledgment that it finished sending the entire file. The last line in the block indicates that the client was waiting for an acknowledgment from the server.
- 14** TRUE (00) indicates that the client received an acknowledgment from the LPD server that the control file was received.
- 15** Confirms that the control file was sent to the LPD server. The job was then terminated.

Figure 19 is a sample LPR trace showing a print request in which the FILTER X option was specified on the LPR command. Since the LPD server does not support this type of filter, it rejects the print request. (For an example of an LPD trace that shows that this job was rejected, see Figure 24 on page 301.) The LPR trace does not show an error because it can send a print request to non-IBM LPDs that support other filters (for example, FILTER X). For detailed information about filters, refer to RFC1179 and to the *z/OS Communications Server: IP Configuration Reference*.

The trace was produced using the following command:

```
LPR test (p TIANNA h 9.67.113.60 filter x TRACE
```

issued through TSO by user ID TCPUSR4.

```

1
EZB0915I Begin "LPR" to printer "TIANNA" at host "9.67.113.6 0"
EZB1057I Loaded translation table from "TCP31S.STANDARD.TCPXLBIN".
EZB0920I Requesting TCP/IP service at 96155 19:22:15
EZB0921I Granted TCP/IP service at 96155 19:22:15
EZB0922I Resolving 9.67.113.60 at 96155 19:22:15
EZB0924I Host 9.67.113.60 name resolved to 9.67.113.60 at 96155 19:22:15
EZB0925I TCP/IP turned on.
EZB0926I Host "MVSA" Domain "TCP.RALEIGH.IBM.COM" TCPIP Service Machine TCP31S
EZB0927I Trying to open with local port 721 to foreign host address 9.67.113.60
EZB0928I Connection open from local port 721 to foreign host address 9.67.113.60
:
2
EZB1009I Data file sent. Port Number = 721. Remote IP Addr = 9.67.113.60
3
EZB1011I Queuing control line "HMSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "PTCPUSR4"
EZB1011I Queuing control line "JTCPUSR4.TEST"
EZB1011I Queuing control line "CMVSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "LTCPUUSR4"
4
EZB1011I Queuing control line "xdfA947MVSA"
EZB1011I Queuing control line "UdfA947MVSA"
EZB1011I Queuing control line "NTCPUSR4.TEST"
EZB0916I Sending command 2 argument: 122 cfA947MVSA Port Number = 721. Remote IP Addr = 9.67.113.60
EZB0917I Command successfully sent Port Number = 721. Remote IP Addr = 9.67.113.60
:
5
EZB1018I Control file sent Port Number = 721. Remote IP Ad dr = 9.67.113.60

```

Figure 19. Example of LPR trace with filter x option

Following are short descriptions of the numbered items in the trace:

- 1** Indicates that the print request was issued to a printer named TIANNA at IP address 9.67.113.60.
- 2** Indicates that the data file was sent. The error was not recognized until the LPD server tried to process the print job. (See Figure 24 on page 301.)
- 3** Indicates control commands sent to the LPD server. For details about these commands, refer to RFC1179.
- 4** Represents the name of the data file. The character string xdf indicates that the x filter was used.
- 5** Indicates that the control file was sent to the LPD server. The job was then terminated.

Figure 20 is a sample showing a print request using the following command `lpr test (p njeSOT0 host MVSA` without the TRACE option. The output shows an error because the printer name was not entered entirely in capital letters.

```
1
EZB1006E Host MVSA did not accept printer name njeSOT0.
      Port Number = 721 Remote IP Addr = 9.67.113.60

2
EZB1049E Send printer command did not receive ACK. ACK message = .
      Port = 721. Remote IP Addr = 9.67.113.60
```

Figure 20. Example of LPR output with unknown printer

Following are short descriptions of the numbered items in the trace.

- 1** Indicates that a SERVICE statement for a printer named njeSOTO did not exist in the LPD server configuration file.
- 2** Indicates that the LPD server did not send a positive response to the LPR client. The job was then terminated.

Figure 21 on page 293 is a sample LPR trace output produced with the following command the JNUM option and variable, along with the LANDSCAPE and TRACE options:

```
lpr test (p TIANNA host 9.67.113.60 JNUM 111 LANDSCAPE TRACE
```

The trace output shows the scanning that occurred to identify the first available port.

```

1
EZB0988I PostScript program is 635 bytes
EZB0915I Begin "LPR" to printer "TIANNA" at host "9.67.113.60"
EZB1057I Loaded translation table from "TCP31S.STANDARD.TCPXLBIN".
EZB0920I Requesting TCP/IP service at 96155 19:35:12
EZB0921I Granted TCP/IP service at 96155 19:35:12
EZB0922I Resolving 9.67.113.60 at 96155 19:35:12
EZB0924I Host 9.67.113.60 name resolved to 9.67.113.60 at 96155 19:35:12
EZB0925I TCP/IP turned on.
EZB0926I Host "MVSA" Domain "TCP.RALEIGH.IBM.COM" TCPIP Service Machine TCP31S

2
EZB0927I Trying to open with local port 721 to foreign host a ddress 9.67.113.60
EZB0927I Trying to open with local port 722 to foreign host address 9.67.113.60
EZB0927I Trying to open with local port 723 to foreign host address 9.67.113.60
EZB0927I Trying to open with local port 724 to foreign host address 9.67.113.60

3
EZB0928I Connection open from local port 724 to foreign host address 9.67.113.60

4
EZB0961I Control file name is cfa111MVSA
EZB0962I Data file name is dfa111MVSA Port Number = 724. Remote I P Addr = 9.67.113.60
EZB0916I Sending command 2 argument: TIANNA Port Number = 724. Remote IP Addr = 9.67.113.60
:
EZB1009I Data file sent. Port Number = 724. Remote IP Addr = 9.67.113.60
EZB1011I Queuing control line "HMVSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "PTCPUSR4"
EZB1011I Queuing control line "JTCPUSR4.TEST"
EZB1011I Queuing control line "CMVSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "LTCPUUSR4"

5
EZB1011I Queuing control line "fdfa111MVSA"
EZB1011I Queuing control line "Udfa111MVSA"
EZB1011I Queuing control line "NTCPUSR4.TEST"
EZB0916I Sending command 2 argument: 122 cfa111MVSA Port Number = 724.

```

Figure 21. Example of LPR trace with JNUM, LANDSCAPE, and TRACE options

Following are short descriptions of the numbered items in the trace:

- 1** Indicates that the LPR client inserted a landscape header, written in postscript, at the beginning of the data file.
- 2** Indicates that the LPR client was attempting to use the first available client port. The port range for the LPR client is 721 through 731. If no ports are available, an error message is displayed.
- 3** Indicates that a connection was opened using port 724.
- 4** Indicates that the value specified for JNUM (111) was used to build the control and data file names.
- 5** Indicates the name of the file containing the three-digit job number that was used with the file name sent to the print server.

Following is a clipping of the header that was inserted into the data file. For more information about header files, refer to *z/OS Communications Server: SNA Customization*.

```

%!PS-Adobe-2.0
614 25 translate 90 rotate .88 .76 scale /n 1 def /fs 10 def /ls 11. 2 def /ld 1

```

Figure 22 on page 294 is a sample of LPR trace output for the following command with the XLATE option:

```
LPR test (p TIANNA h MVSA trace xlate GXS
```


In this sample, the server was not running, so the connection was not established.
For detailed information about using and creating your own translate tables, refer to
z/OS Communications Server: SNA Customization

```
EZB0915I Begin "LPR" to printer "TIANNA" at host "MVSA"  
1  
EZB1057I Loaded translation table from "TCPUSR4.GXS.TCPXLBIN" .  
EZB0920I Requesting TCP/IP service at 96155 20:04:14  
EZB0921I Granted TCP/IP service at 96155 20:04:15  
2  
EZB0922I Resolving MVSA at 96155 20:04:15  
3  
EZB0924I Host MVSA name resolved to 9.67.113.60 at 96155 20:0 4:17  
EZB0925I TCP/IP turned on.  
EZB0926I Host "MVSA" Domain "TCP.RALEIGH.IBM.COM" TCPIP Service Machine TCP31S  
EZB0927I Trying to open with local port 721 to foreign host address 9.67.113.60  
4  
EZB1051E Failed to Open connection to Port Number = 515. Return  
Code = -1. Error Number = 61. Port Number = 721.  
Remote IP Addr = 9.67.113.60
```

Figure 22. Example of LPR trace with XLATE option

Following are short descriptions of the numbered items in the trace:

- 1 Indicates the name of the translation table used by the LPR client. To avoid problems such as errors and data corruption, be sure that the LPD server is using the equivalent code pages.
- 2 Indicates the time the LPR client started trying to resolve the specified host name. The LPR client checks the name server table, the site, and address information files to resolve the host name.
- 3 Indicates the amount of time the LPR client took to resolve the specified host name. To reduce the amount of time, use the host IP address instead of the host name.
- 4 Indicates that the connection was not established. (In this sample, the LPD server was not running.) For a list of error numbers and their definitions, refer to *z/OS Communications Server: IP and SNA Codes*.

LPD server traces

This section includes information on activating LPD server traces. It also provides samples of LPD trace output with explanations of selected messages.

Activating server traces

You can activate the tracing facilities within the LPD server in any of the following ways:

- Include the TRACE parameter in the LPSERVE PROC statement in the LPD server cataloged procedure.

Be sure that a slash (/) precedes the first parameter and that each parameter is separated by a blank. For example:

```
//LPSERVE PROC MODULE='LPD',PARMS='/TRACE'
```

- Enter the command **SMSG procname**, where *procname* is the name of the procedure used to start the LPD server.
- Specify the DEBUG statement in the LPD configuration file, LPDDATA.

Server trace output

LPD server traces go to the SYSPRINT data set. You can also define a DD card in the LPD cataloged procedure to write output to another data set. This section contains some samples of LPD server trace output.

Figure 23 is a sample of an LPD trace invoked by specifying the DEBUG option in the LPD configuration file, LPDDATA.

```
EZB0832I
EZB0621I LPD starting with port 515
EZB0679I Allocated ObeyBlock at 00005B70
EZB0679I Allocated ObeyBlock at 00005B60
EZB0679I Allocated ObeyBlock at 00005B50
EZB0628I Allocated PrinterBlock at 000058C0
EZB0629I prt1 added.
EZB0641I Service prt1 defined with address
EZB0628I Allocated PrinterBlock at 00005630
EZB0629I PRT1 added.
EZB0641I Service PRT1 defined with address
EZB0628I Allocated PrinterBlock at 000053A0
EZB0629I TIANNA added.
EZB0641I Service TIANNA defined with address
EZB0628I Allocated PrinterBlock at 00005110
EZB0629I PRT2 added.
EZB0641I Service PRT2 defined with address
EZB0628I Allocated PrinterBlock at 000B1D40
EZB0629I njesoto added.
EZB0641I Service njesoto defined with address
EZB0628I Allocated PrinterBlock at 000B1AB0
EZB0629I rda added.
EZB0686I Host "9.37.33.159" resolved to 9.37.33.159. Printer name is "lpt1".
EZB0641I Service rda defined with address
EZB0628I Allocated PrinterBlock at 000B1820
EZB0629I POST added.
```

Figure 23. Example of LPD trace specified with the DEBUG option (Part 1 of 5)

```

2
EZB0686I Host "9.67.105.55" resolved to 9.67.105. 55.  Printer name is "LPT2".
2
EZB0641I Service POST defined with address
EZB0697I   ...End of Printer chain...
EZB0626I Allocated ConnectionBlock at 00147E08
3
EZB0627I Passive open on port 515
EZB0705I 06/03/96 18:49:15
EZB0834I Ready
4
EZB0789I GetNextNote with ShouldWait of TRUE
.
.
5
EZB0790I GetNextNote returns.  Connection 1 NotificationConnection state changed (8681)
5
EZB0779I New connection state Open (8673) on connection 1 with reason OK.
5
EZB0782I Connection open.  Reading command.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0711I New command 2 data "2".
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
6
EZB0754I New subcommand 3 operands "7434 dfA827MV SA".
EZB0723I Allocated StepBlock at 000B1320
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Trying to open (8676) on connection 2 with reason OK.
EZB0626I Allocated ConnectionBlock at 0015BE08
7
EZB0627I Passive open on port 515
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Open (8673) on connection 2 with reason OK.
EZB0782I Connection open.  Reading command.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 2 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 2
EZB0711I New command 4 data "4".
EZB0708I FSend of response sent
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 2 Notification FSend response (8692)
EZB0763I Closing connection 2
EZB0789I GetNextNote with ShouldWait of TRUE

```

Figure 23. Example of LPD trace specified with the DEBUG option (Part 2 of 5)

```

EZB0790I GetNextNote returns. Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Receiving only (8674) on connection 2 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
8
EZB0754I New subcommand 2 operands "153 cfA827MVS A".
EZB0723I Allocated StepBlock at 000B1168
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Connection closing (8670) on connection 2 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Sending only (8675) on connection 1 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification FSend response (8692)
EZB0763I Closing connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Connection closing (8670) on connection 1 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Nonexistent (8672) on connection 1 with reason OK.
EZB0772I End Connection 1 for OK.

```

Figure 23. Example of LPD trace specified with the DEBUG option (Part 3 of 5)

```

9
EZB0776I Released StepBlock at 000B1320
9
EZB0719I Allocated JobBlock at 00147798
9
EZB0723I Allocated StepBlock at 000B1320
10
EZB0716I Job 827 received prt1 MVSA
11
EZB0734I Job 827 added to work queue
12
EZB0716I Job 827 scheduled prt1 MVSA
EZB0776I Released StepBlock at 000B1168
EZB0777I Released ConnectionBlock at 0014AE08
EZB0824I ProcessWork starting on job queue
13
EZB0731I Work Queue start
13
EZB0732I $      827 JOBstartPRINTING
EZB0733I      Work Queue end
EZB0825I      Job 827 for prt1 dispatched in state JOBstartPRINTING
EZB0716I Job 827 printing prt1 MVSA
EZB0827I ProcessWork end with queue
EZB0731I      Work Queue start
14
EZB0732I $      827 JOBcontinuePRINTING
EZB0733I      Work Queue end
EZB0789I GetNextNote with ShouldWait of FALSE
EZB0824I ProcessWork starting on job queue
EZB0731I      Work Queue start
EZB0732I $      827 JOBcontinuePRINTING
EZB0733I      Work Queue end
EZB0825I      Job 827 for prt1 dispatched in state JOBcontinuePRINTING
      flpNewBlock: State first call IsAtEof FALSE
15
      flpNewBlock: State build      IsAtEof FALSE
      flpNewBlock: State check last IsAtEof FALSE
:
:
      flpNewBlock: State check last IsAtEof FALSE
      flpNewBlock: State build      IsAtEof FALSE
:
:
EZB0825I      Job 827 for prt1 dispatched in state JOBcontinuePRINTING
:
:
      flpNewBlock: State build      IsAtEof TRUE
      flpNewBlock: State check last IsAtEof TRUE
EZB0827I ProcessWork end with queue
EZB0731I      Work Queue start
EZB0732I $      827 JOBcontinuePRINTING
EZB0733I      Work Queue end
EZB0789I GetNextNote with ShouldWait of FALSE
EZB0824I ProcessWork starting on job queue

```

Figure 23. Example of LPD trace specified with the DEBUG option (Part 4 of 5)

```

EZB0731I      Work Queue start
EZB0732I $      827 JOBcontinuePRINTING
EZB0733I      Work Queue end
EZB0825I      Job 827 for prt1 dispatched in state JOBcontinuePRINTING
EZB0827I ProcessWork end with queue
EZB0731I      Work Queue start
EZB0732I $      827 JOBfinishPRINTING
EZB0733I      Work Queue end
EZB0789I GetNextNote with ShouldWait of FALSE
EZB0824I ProcessWork starting on job queue
EZB0731I      Work Queue start
16
EZB0732I $      827 JOBfinishPRINTING
EZB0733I      Work Queue end
EZB0825I      Job 827 for prt1 dispatched in state JOBfinishPRINTING
17
EZB0716I Job 827 sent prt1 MVSA
17
EZB0769I Job 827 removed from work queue
EZB0751I Released StepBlock at 000B1320
17
EZB0716I Job 827 purged prt1 MVSA
EZB0771I Released JobBlock at 00147798
18
EZB0827I ProcessWork end with queue
EZB0731I      Work Queue start
EZB0733I      Work Queue end
19
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 0 Notification Connection state changed (8681)
20
EZB0779I New connection state Nonexistent (8672) on connection 0 with reason OK.
20
EZB0772I End Connection 0 for OK.
EZB0777I Released ConnectionBlock at 00147E08
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Nonexistent (8672) on connection 2 with reason OK.
EZB0772I End Connection 2 for OK.
EZB0777I Released ConnectionBlock at 0014DE08
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 3 Notification Connection state changed (8681)
EZB0779I New connection state Trying to open (8676) on connection 3 with reason OK.
EZB0626I Allocated ConnectionBlock at 00147E08
EZB0627I Passive open on port 515
EZB0789I GetNextNote with ShouldWait of TRUE
:
21
EZB0790I GetNextNote returns. Connection -48 Notification Other external interrupt received (8688)
21
EZB0622I Terminated by external interrupt

```

Figure 23. Example of LPD trace specified with the *DEBUG* option (Part 5 of 5)

Following are short descriptions of the numbered items in the trace:

- 1** Indicates that a control block was allocated for each service defined in the LPD configuration file. TIANNA is the name of one of the local printers.
- 2** Indicates that the remote printer, LPT2, was defined in a *SERVICE* statement with the name POST. LPT2 has the IP address 9.67.105.55.
- 3** Indicates that the LPD server listened on port 515 and that port 515 was opened.
- 4** Indicates that the LPD server waited for work.

- 5** Indicates that a connection was opened for an incoming LPR client and that the LPD server was receiving a command from that client.
- 6** Indicates that a subcommand was received from an LPR client. The subcommand indicates LPD was receiving a data file named dfA827MVSA, containing 7434 bytes of data. For details on commands and subcommands, refer to RFC117.
- 7** Indicates that the LPD server had a passive open connection on the restricted LPD port, 515.
- 8** Indicates that the LPD server was receiving a control file named cfA827MVSA, containing 153 bytes of data.

Note: Data files use the naming convention of dfx. Control files use the naming convention cfx.

- 9** Indicates the control blocks that were allocated and released as files were received and processed. Control blocks are used primarily by IBM support for debugging purposes, in coordination with dumps.
- 10** Indicates that all data files for a particular job were received.

Note: Job number 827 is a three-digit job number generated by the LPR client.

- 11** Indicates that job 827 was added to this print queue. The LPD server maintains a work queue of jobs.
- 12** Indicates that job 827 was scheduled to be spooled to the output queue.
- 13** Indicates that the LPD server was processing print jobs from the work queue, and started sending print data to the JES output queue. The message JOBstartPRINTING does not mean that the file is physically printing.
- 14** Indicates that data was being sent for output. Depending on the size of the file, you may see this status many times for a single job.
- 15** Indicates checking for the end of the file as it is being processed. The number of IsAtEof entries depends on the data and size of the file.
- 16** Indicates that all data was processed and placed in the output queue.
- 17** Indicates that job 827 was completely processed by the LPD server and removed from the print queue, prt1, on host MVSA. Temporary data sets and control blocks for this job were also erased or released.
- 18** Indicates that the LPD server completed the jobs in that queue, and will scan the work queue again.
- 19** Indicates that the LPD server was waiting for more work to do.
- 20** Indicates that the LPR-to-LPD connection was closed normally.
- 21** Indicates that someone stopped the LPD server normally.

Figure 24 on page 301 is a sample of LPD trace output showing that job 947 failed to print because the client passed a filter that was not supported by the LPD server. In cases such as these, you can lose printouts. In this case, the LPD trace showed why, but the LPR trace did not show an error. (See Figure 19 on page 291 for the corresponding LPR trace output.)

```

EZB0831I IBM MVS LPD Version V2R10 on 05/05/98 at 19:21:46
EZB0832I
EZB0621I LPD starting with port 515
.:
EZB0628I Allocated PrinterBlock at 000053A0
EZB0629I     TIANNA added.
EZB0641I Service TIANNA defined with address
.:
EZB0627I Passive open on port 515
EZB0705I 06/03/96 19:21:47
EZB0834I Ready
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.Connection 0 Notification Connection state changed(8681)
EZB0779I New connection state Trying to open (8676) on connection 0 with reason OK.
EZB0626I Allocated ConnectionBlock at 0014AE08
EZB0627I Passive open on port 515
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.Connection 0 Notification Connection state changed(8681)
EZB0779I New connection state Open (8673) on connection 0 with reason OK.
EZB0782I Connection open. Reading command.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0627I Passive open on port 515
.:
1
EZB0754I New subcommand 3 operands "333819 dfA947MV SA".
.:
2
EZB0754I New subcommand 2 operands "122 cfA947MVSA" .
.:
EZB0776I Released StepBlock at 000B1438
EZB0719I Allocated JobBlock at 00147798
EZB0723I Allocated StepBlock at 000B1438
3
EZB0716I Job 947 received TIANNA MVSA
3
EZB0734I Job 947 added to work queue
3
EZB0716I Job 947 scheduled TIANNA MVSA
EZB0776I Released StepBlock at 000B1280
EZB0777I Released ConnectionBlock at 0014AE08
EZB0824I ProcessWork starting on job queue
EZB0731I     Work Queue start
EZB0732I $       947 JOBstartPRINTING
EZB0733I     Work Queue end
EZB0825I     Job 947 for TIANNA dispatched in state JOBstartPRINTING
EZB0716I Job 947 printing TIANNA MVSA

```

Figure 24. Example of an LPD server trace of a failing job (Part 1 of 2)

```

4
EZB0801I Filter "x" not supported. Job abandoned.
EZB0827I ProcessWork end with queue
EZB0731I      Work Queue start
EZB0732I $      947 JOBfinishPRINTING
EZB0733I      Work Queue end
EZB0789I GetNextNote with ShouldWait of FALSE
EZB0790I GetNextNote returns.Connection 0 Notification Connection state changed(8681)
EZB0779I New connection state Connection closing (8670) on connection 0 with reason OK.
EZB0824I ProcessWork starting on job queue
EZB0731I      Work Queue start
EZB0732I $      947 JOBfinishPRINTING
EZB0733I      Work Queue end
5
EZB0825I      Job 947 for TIANNA dispatched in state JOBfinishPRINTING
EZB0716I Job 947 sent TIANNA MVSA
6
EZB0769I Job 947 removed from work queue
EZB0751I Released StepBlock at 000B1438
7
EZB0716I Job 947 purged TIANNA MVSA
EZB0771I Released JobBlock at 00147798
EZB0827I ProcessWork end with queue
EZB0731I      Work Queue start
EZB0733I      Work Queue end
:
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection -48 Notification Other external interrupt received (8688)
EZB0622I Terminated by external interrupt

```

Figure 24. Example of an LPD server trace of a failing job (Part 2 of 2)

Following are short descriptions of the numbered items in the trace:

- 1** Indicates that the LPD server received a command indicating the byte size and name of a data file sent by an LPR client.
- 2** Indicates that the LPD server received a command indicating the byte size and name of a control file sent by an LPR client.
- 3** Indicates that print job 947 was received, placed in the print queue named TIANNA on host MVSA, and was scheduled to be processed.
- 4** Indicates that the LPD server did not support filter x and discarded the print job.
- 5** Indicates that the job was finished. The flag JOBfinishPRINTING indicates the job is to be removed from the work queue and purged.
- 6** Indicates that the job was removed from the work queue and that the control blocks were released.
- 7** Indicates that the job was purged.

Figure 25 on page 304 is a sample of an LPD trace output generated by specifying the DEBUG statement in the LPD configuration file (LPDDATA). This sample shows that an LPR client issued a request, through an LPD server, to a printer defined as a remote server. (The LPD server acted as an LPR client by sending the request to a remote server.) Since the remote server was not running, the print job was purged.

Initially, the LPR client was unaware that the server was not running because the LPD server correctly acknowledged receipt of the data files and control files. Furthermore, the LPR trace did not indicate any problems. However, if you specify

the option FAILEDJOB MAIL on the SERVICE statement for the remote printer, notification is sent to the user ID of the LPR client. For notification to be sent, Simple Mail Transfer Protocol (SMTP) must be running.

Note: The FAILEDJOB DISCARD option is the default.

The command **LPR lpd.config (p SOTO h MVS7** was used to generate the trace output. SOTO is the name of the printer specified on the SERVICE statement, and MVS7 is the host on which the LPD server is running.

```

1
EZB0831I IBM MVS LPD Version V2R10
  on 05/05/98 at 19 :50:58
EZB0832I
EZB0621I LPD starting with port 515
EZB0679I Allocated ObeyBlock at 00005B70
EZB0679I Allocated ObeyBlock at 00005B60
EZB0679I Allocated ObeyBlock at 00005B50
EZB0628I Allocated PrinterBlock at 000058C0
EZB0629I prt1 added.
EZB0641I Service prt1 defined with address
EZB0628I Allocated PrinterBlock at 00005630
EZB0629I PRT1 added.
EZB0641I Service PRT1 defined with address
EZB0628I Allocated PrinterBlock at 000053A0
EZB0629I TIANNA added.
EZB0641I Service TIANNA defined with address
EZB0628I Allocated PrinterBlock at 00005110
EZB0629I PRT2 added.
EZB0641I Service PRT2 defined with address
EZB0628I Allocated PrinterBlock at 000B1D40
EZB0629I njesoto added.
EZB0641I Service njesoto defined with address
EZB0628I Allocated PrinterBlock at 000B1AB0
EZB0629I SOTO added.

2
EZB0686I Host "9.37.34.39" resolved to 9.37.34.39. Printer name is "lpt1".

3
EZB0641I Service SOTO defined with address
EZB0628I Allocated PrinterBlock at 000B1820
EZB0629I POST added.
EZB0686I Host "9.67.105.55" resolved to 9.67.105.55. Printer name is "LPT2".
EZB0641I Service POST defined with address
EZB0697I ...End of Printer chain...
EZB0626I Allocated ConnectionBlock at 00147E08
EZB0627I Passive open on port 515
EZB0705I 06/05/96 19:50:00
EZB0834I Ready
EZB0789I GetNextNote with ShouldWait of TRUE
:
EZB0782I Connection open. Reading command.
EZB0799I Reading additional data on 1
:
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1

4
EZB0754I New subcommand 3 operands "14221 dfA502MVS 7".
EZB0723I Allocated StepBlock at 000B1438
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0789I GetNextNote with ShouldWait of TRUE
:
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1

```

Figure 25. Example of an LPD server trace for a remote print request (Part 1 of 3)

```

5
19:50:48 EZB0754I New subcommand 2 operands "134 cfA502MVS7" .
19:50:48 EZB0723I Allocated StepBlock at 000B1280
19:50:49 EZB0789I GetNextNote with ShouldWait of TRUE
:
:
6
EZB0763I Closing connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Connection closing (8670) on connection 1 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Nonexistent (8672) on connection 1 with reason OK.
EZB0772I End Connection 1 for OK.
EZB0719I Allocated JobBlock at 00147798
7
EZB0716I Job 502 received SOTO MVS7
7
EZB0734I Job 502 added to work queue
7
EZB0716I Job 502 scheduled SOTO MVS7
EZB0777I Released ConnectionBlock at 0014AE08
EZB0824I ProcessWork starting on job queue
EZB0731I Work Queue start
8
EZB0732I $502 JOBstartSENDING
EZB0733I Work Queue end
EZB0825I Job 502 for SOTO dispatched in state JOBstartSENDING
EZB0626I Allocated ConnectionBlock at 0014AE08
9
EZB0820I Trying to open with local port 721
9
EZB0716I Job 502 opening SOTO MVS7
10
EZB0769I Job 502 removed from work queue
EZB0827I ProcessWork end with queue
EZB0731I Work Queue start
EZB0733I Work Queue end
EZB0789I GetNextNote with ShouldWait of TRUE
:
:
EZB0790I GetNextNote returns.Connection 1 Notification Connection state changed(8681)
11
EZB0779I New connection state Nonexistent(8672) on connection 1 with reason
Foreign host did not respond within OPEN
11
EZB0772I End Connection 1 for Foreign host
did not respond within OPEN timeout (8560).
EZB0705I 06/05/96 19:52:22
12
EZB0773I Connection 1 terminated because "Foreign host did not respond within OPEN timeout (8560)"

```

Figure 25. Example of an LPD server trace for a remote print request (Part 2 of 3)

```

13
EZB0744I 748656 HELO MVS7.tcp.raleigh.ibm.com
13
EZB0744I 748656 MAIL FROM:<LPDSRV3@MVSA>
13
EZB0744I 748656 RCPT TO:<TCPUSR4@MVS7.tcp.raleigh.  ibm.com>
13
EZB0744I 748656 DATA
13
EZB0744I 748656 To:<TCPUSR4@MVS7.tcp.raleigh.ibm.com>
13
EZB0744I 748656
13
EZB0744I 748656 Your job to print the files "TCPUSR 4.LPD.CONFIG" on SOTO at MVSA has failed for
13
EZB0744I 748656 this reason: Remote connection
terminated (Foreign host did not respond within
13
EZB0744I 748656 OPEN timeout (8560)).
13
EZB0744I 748656 .
EZB0751I Released StepBlock at 000B1438
EZB0751I Released StepBlock at 000B1280
14
EZB0716I Job 502 purged SOTO MVS7
EZB0771I Released JobBlock at 00147798
EZB0777I Released ConnectionBlock at 0014AE08
EZB0789I GetNextNote with ShouldWait of TRUE
15
EZB0790I GetNextNote returns. Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Nonexistent (8672) on connection2 with reason OK.
EZB0772I End Connection 2 for OK.
EZB0777I Released ConnectionBlock at 0014DE08
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 0 Notification Connection
state changed (8681)
EZB0779I New connection state Trying to open (8676) on connection0 with reason OK.
EZB0626I Allocated ConnectionBlock at 00147E08
EZB0627I Passive open on port 515
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 0 Notification Connection state changed (8681)
EZB0779I New connection state Open (8673) on connection 0 with reason OK.
EZB0782I Connection open. Reading command.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 0 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 0
EZB0711I New command 4 data "4".
EZB0708I FSend of response sent
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 0 Notification FSend response (8692)
EZB0763I Closing connection 0
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 0 Notification Connection state changed (8681)
EZB0779I New connection state Receiving only (8674) on connection0 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection -48 Notification Other external interrupt received (8688)
EZB0622I Terminated by external interrupt

```

Figure 25. Example of an LPD server trace for a remote print request (Part 3 of 3)

Following are short descriptions of the numbered items in the trace:

- 1** Indicates the date and time the LPD server was activated. This information can be compared to the date and time on an LPR trace to assure that both traces were generated for the same incident.

- 2** Indicates the IP address of the host. If the name of the host was specified instead of the IP address, this message would indicate if the IP address of the host was resolved.
- 3** Indicates that the name SOTO was defined on the SERVICE statement for the remote printer, lpt1, which had the address 9.37.34.39.
- 4** Indicates the byte size and the name of the data file sent from the LPR client on host MVS7.
- 5** Indicates the byte size and name of control file sent from the LPR client on host MVS7.
- 6** Indicates that the connection between the LPR client and the LPD server was closing, after the server received the data and control files.
- 7** Indicates that the print job was received, placed in the LPD print queue, represented by SOTO, and scheduled to be sent to its destination.
- 8** Indicates that the LPD server started to send print job 502 to the remote server.

Note: If the printer was local, rather than remote, the message would have read 502 JOBstartPRINTING.
- 9** Indicates that the LPD server, acting as a client, was opening a connection to the remote printer using local port 721.
- 10** Indicates that the LPD server removed the job from its work queue.
- 11** Indicates that the connection to the remote server timed out.
- 12** Indicates that the remote server did not respond to the request to open.
- 13** Indicates that the FAILEDJOB MAIL option was defined under the SERVICE statement and that SMTP was running. The text in these messages was sent to the user ID of the LPR client.
- 14** Indicates that the print job was completely purged.
- 15** Describes additional activity between the LPD server and other clients.

Chapter 8. Diagnosing File Transfer Protocol (FTP) problems

This chapter describes how to diagnose problems with the z/OS CS FTP server and FTP client. If, after reading this chapter, you are unable to solve your problem and you need to call the IBM Software Support Center, see one or both of the following sections for the documentation you need to provide: “Logging FTP server activity” on page 334 and “Documenting FTP client problems” on page 348.

FTP server

This section describes the following topics:

- “Structural overview”
- “Definitions and setup”
- “Error exit codes” on page 310
- “Name considerations for z/OS UNIX FTP” on page 310
- “Translation and data conversion support” on page 311
- “Common z/OS UNIX FTP problems” on page 313
- “Diagnosing FTP server problems with traces” on page 326
- “Logging FTP server activity” on page 334

Structural overview

The z/OS model for the FTP server includes a daemon process and a server process. The daemon process starts when you start your cataloged procedure (for example, START FTPD) and it listens for connection requests on a specific port. The port is the well-known port 21 unless otherwise specified. For methods of choosing a different port number, see the information about configuring ETC.SERVICES and configuring the FTPD cataloged procedure in the *z/OS Communications Server: IP Configuration Guide*. When the daemon accepts an incoming connection, it creates a new process (server’s address space) for the FTP server, which handles the connection for the rest of the FTP login session. Each login session has its own server process.

The server process inherits the accepted connection from the daemon process. This connection is called the control connection. The server receives commands from the client and sends replies to the client using the control connection. The control connection port is the same as the daemon’s listening port.

The client and server use a different connection for transferring data; this connection is called the data connection. By default, the data port is one less than the control connection port. For example, if the control connection port is 21, then the data port is 20. An FTP client can override the default data port by directing the server to run in passive mode. In passive mode, the server uses an ephemeral port for the data port. Passive mode is requested by firewall friendly clients and by clients initiating three-way data transfers.

Definitions and setup

This section describes the definitions and setup for the FTP server.

Start procedure

The start procedure for the FTP server is EZAFTPAP (alias FTPD) in the *hlq.SEZAINST* data set. Changes might be necessary to customize the start procedure for your MVS host system. The following should be kept in mind for the FTP server start procedure:

- The library containing FTPD and FTPDNS must be APF authorized and must be either in the MVS link list or included on the STEPLIB DD statement.
- The C run-time libraries are needed for FTPD and FTPDNS. They must be APF authorized. If the C run-time library is not in the MVS link list, it must be included on the STEPLIB DD statement.
- If the FTP server will be used for SQL queries, the DB2[®] DSNLOAD library must be APF authorized and must be either in the MVS link list or included on the STEPLIB DD statement.
- Several start options are available for the FTP server. If specified in the start procedure, these values override the default values for the FTP server and any values specified in the FTP.DATA data set.

For more information about the FTP server start procedure, refer to the *z/OS Communications Server: IP Configuration Reference*.

FTP.DATA data set

The FTP.DATA data set is an optional data set that allows the FTP server configuration parameters to be customized. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about the FTP.DATA data set.

TCPIP.DATA data set

The TCPIP.DATA data set provides information to the FTP server, such as the high-level qualifier to be used for configuration data sets, whether messages are to be written in uppercase or mixed case, and which DBCS translation tables are to be used. For more information about the TCPIP.DATA data set, refer to the *z/OS Communications Server: IP Configuration Reference*.

Error exit codes

z/OS UNIX FTP uses the following error exit codes:

- | | |
|-----------|--|
| 12 | Daemon initialization failed; unable to accept an incoming connection. An EZY message identifying the specific problem is sent to syslogd. |
| 24 | The client sessions initialization terminated because the FTP server load module cannot be loaded or executed. Message EZYFT53E is sent to syslogd. |
| 28 | Daemon initialization was terminated because the IBM TCP/IP is not enabled in the IFAPRDxx parmlib member. Message EZYFT54E is sent to syslogd and the operator console. |

Name considerations for z/OS UNIX FTP

This section explains the MVS and HFS naming conventions.

MVS naming conventions

MVS data set names used with all FTP commands sent to the z/OS UNIX FTP server must meet MVS data set naming conventions as follows:

- Data set names can be no longer than 44 characters.

If the path name parameter sent with an FTP command is not enclosed in single quotation marks, the path name is appended to the current working directory to

create the data set name. The combination of the current working directory and the path name cannot be longer than 44 characters. The current working directory can be displayed by issuing the PWD command.

- Each qualifier in a data set name, or each member name for a partitioned data set, must conform to the following:
 - It must be no longer than eight characters.
 - It must begin with a letter or the special characters \$, @, or #.
 - It can contain only numbers, letters, or the special characters \$, @, #, -, or }.
- Generation data group data set names must be in the format *gdg_name(generation_level)*. The *generation_level* is either 0, +*nn*, or -*nn*, where *nn* is the generation number. For example, the GDG data set MYGDG could be specified as MYGDG(0) for the current generation level, MYGDG(-1) for the next to the latest generation level, or MYGDG(+1) for the new generation level.

HFS naming conventions

The following list describes some naming conventions you should know about when using HFS files with the z/OS UNIX FTP server:

- The HFS name is case-sensitive.
- If a name begins with a single quotation mark, specify QUOTESOVERRIDE FALSE in FTP.DATA, or use the SITE NOQUOTESOVERRIDE command.
- Names may contain imbedded blanks for special characters. Be aware that:
 - The subcommand is interpreted as:
`<ftp_subcommand><one blank space><pathname><new_line>`
 - Some FTP clients may truncate trailing blanks.
- The LIST and NLST subcommands, including all client subcommands that invoke the NLST subcommand, such as MGET or MDELETE, require special handling for certain special characters. For more information, refer to *z/OS Communications Server: IP User's Guide and Commands*.
- The START and SITE parameters have additional restrictions on the path name used with SBDATACONN. Refer to *z/OS Communications Server: IP Configuration Reference* and *z/OS Communications Server: IP User's Guide and Commands*.
- When specifying a z/OS UNIX FTP subcommand with a file name containing special characters, some FTP clients may truncate trailing blanks, compress multiple internal blanks, or interpret special characters to have special meanings. Unique specification of the file name such as enclosing in double or single quotation mark, or escaping special characters, may be necessary to get the client to send the file name to the server correctly. Refer to your client documentation to see if this is necessary.

Translation and data conversion support

This section describes translation and data conversion support for the FTP server.

Double-byte character set (DBCS) support

If you enter quote type b <n> at the client and if the DBCS translate table has not been loaded, the following reply is displayed:

```
504-Type not Supported. Translation table not loaded.
```

Do one or both of the following:

- Check the LOADDBCSTABLES statement in the TCPIP.DATA configuration file. If the statement wraps to the next line, parameters on the continued line are ignored. If all the parameters for the LOADDBCSTABLES statement do not fit on one line, use multiple LOADDBCSTABLES statements.
- Check the precedence order for TCPIP.DATA to ensure that the file being used contains the LOADDBCSTABLES statement or statements. Be aware that the location of TCPIP.DATA statements can be influenced in multiple ways, for example, by a GlobalTCPIPData specification or the RESOLVER_CONFIG environment variable. Refer to the *z/OS Communications Server: IP Configuration Guide* for the TCPIP.DATA search order.

Single-byte character (SBCS) support

Data conversion occurs for single-byte data on the data connection when ENCODING=SBCS is in effect and the data type is ASCII. For more information, refer to the FTP.DATA statement ENCODING in the *z/OS Communications Server: IP Configuration Reference* and the SITE ENCODING command in the *z/OS Communications Server: IP User's Guide and Commands*.

If you choose SBDATACONN as a statement in the FTP.DATA file or with the SITE SBDATACONN command, the FTP server builds a translation table using the codepages specified by SBDATACONN. If you receive the following reply to the SITE command, ask for a trace of the server with the UTL option to determine which characters cannot be translated.

200 Some characters cannot be translated between *codepage_1* and *codepage_2*.

If none of the untranslatable characters appear in the data, the data transfers are not affected. If, however, one of the untranslatable characters does appear, the data transfer will fail and the client will receive the following reply:

557 Data contains codepoints that cannot be translated.

You can avoid the failure if you specify a substitution character to replace non-translatable characters. For details on how to ask for character substitution, refer to SBSUB and SBSUBCHAR as FTP.DATA statements in the *z/OS Communications Server: IP Configuration Reference* and as parameters on the SITE command in *z/OS Communications Server: IP User's Guide and Commands*. If substitution occurs during the transfer, the client will receive the following reply:

250 One or more characters were substituted during the transfer.

When substitution occurs at the destination of a data transfer, a subsequent transfer of the resulting data will not produce an exact copy of the original. For example, if you put a file to the server and one or more characters are substituted, the untranslatable characters are overlaid in the server copy with the substitution character. You cannot restore the original file by getting it from the server.

Multibyte character set (MBCS) support

Data conversion occurs for multibyte data on the data connection when ENCODING=MBCS is in effect and the data type is ASCII. For more information, refer to the FTP.DATA statement ENCODING in the *z/OS Communications Server: IP Configuration Reference* and the SITE ENCODING command in the *z/OS Communications Server: IP User's Guide and Commands*.

If you choose ENCODING=MBCS, you must specify MBDATACONN with a statement in the FTP.DATA file or with the SITE MBDATACONN command to name

the codepages for the multibyte data transfer. If you attempt an ASCII data transfer with ENCODING=MBCS and no MBDATACONN specified, the client will receive the following reply:

504 Multibyte encoding set but codepages are not defined.

If the multibyte data that you transfer has codepoints that cannot be translated, the transfer fails and the client will receive the following reply:

557 Data contains codepoints that cannot be translated.

You can determine which bytes of the data cannot be translated by repeating the transfer with the DUMP 42 extended trace option active at the server.

Common z/OS UNIX FTP problems

This section describes some common z/OS UNIX FTP problems.

FTP daemon initialization problems

The following problems may be encountered when the FTP daemon is initialized.

No "Initialization Complete" message: If the EZY2702I Server-FTP Initialization completed at ... message does not appear on the system console within a few minutes after starting the FTP daemon, verify that the daemon background job is still running. For example, if you started FTP with a procedure called FTPD, you can use the D A,L command to see if the job FTPD1 is active.

If the background job daemon job is running (for example, FTPD1), verify that TCP/IP is running. If it is not, start TCP/IP. The FTP initialization will complete when TCP/IP starts.

If the background daemon job is not running, check the system console for nonzero exit codes from the background job. Look for messages in message or trace output from syslogd for an EZY error message from FTP. Following are the possible exit codes and the appropriate responses:

- 0012

FTP is unable to use the port specified for the control connection. Look in the syslogd messages for the specific reason. Possible errors include the following:

- EZYFT13E bind error...Operation not permitted

Ensure that FTP has BPX.DAEMON authority.

- EZYFT13E bind error...Address already in use

Ensure that FTP is trying to use the correct port. The FTP server trace with the INT option indicates the port the daemon expects to use. If this is the correct port, you can use the TSO NETSTAT CONN command to determine the job that is currently using that port.

- EZYFT13E bind error...Permission denied

Ensure that the port you want FTP to use has been reserved for the FTP background job name. For example, if your start procedure is called FTPD and you want FTP to use port 21, the PORT statement in your *hlq.PROFILE.TCPIP* data set must specify 21 TCP FTPD1.

- 0028

This FTP daemon is not available because the IBM TCP/IP is not enabled.

Incorrect configuration values: If you experience incorrect configuration values, check the following:

- Look in syslogd output to verify that configuration values are coming from your intended FTP.DATA file and that no errors were encountered.
- Determine whether your FTP.DATA file has sequence numbers. If it does, any statement with an optional parameter omitted will pick up the sequence number as the parameter value.

For example, the BLKSIZE statement has an optional parameter size. If you specify the size, the sequence number is ignored. If you do not specify the size, the system assumes the sequence number is the size, causing an error.

FTP daemon not listening on expected port: If the problem is that the daemon is not listening on the expected port, verify that the correct port number is specified. Following is the preference order for a port number:

1. PORT start parm
2. /etc/services
3. *hlq*.ETC SERVICES
4. A default port number of 21

AUTOLOG does not start the FTP daemon: If your start procedure name has fewer than eight characters, ensure that the AUTOLOG and PORT statements in the *hlq*.PROFILE.TCPIP data set specify the FTP background job name. For example, if your start procedure is called FTPD, your *hlq*.PROFILE.TCPIP data set should specify FTPD1, as shown in the following examples:

```
AUTOLOG
  FTPD JOBNAME FTPD1
ENDAUTOLOG
PORT
  20 TCP OMVS NOAUTOLOG ;FTP data port
  21 TCP FTPD1           ;FTP control port
```

FTP server abends

If the FTP server abends, check the following:

- S683 or U4088 abend validating user ID or password.
 - Ensure that the sticky bit has been turned on for the files /usr/sbin/ftpd and /usr/sbin/ftpdns.
 - Ensure that the FTPD and FTPDNS modules reside in an APF authorized partitioned data set, which is specified in the MVS linklist.
 - Ensure that all programs loaded into the FTP address space are APF authorized and are marked as controlled. This means that any FTP user exits, the SQL load library, and the loaded run-time library need to be marked as controlled, using the RACF RDEFINE command. For more information, refer to *z/OS UNIX System Services Planning*, or refer to the RACF publications.

FTP session problems

The following sections describe some common FTP session problems.

Connection terminated by the server after user enters user ID: The system console may display one of the following nonzero exit codes from the FTP server address space:

- 0012
This exit code indicates a socket error. See the syslogd messages for the specific error.
- 0024

This exit code indicates that the system was unable to load the server load module /usr/sbin/ftpdns. Ensure that the symbolic link or links for ftpdns are correct, that ftpdns exists in the HFS and that the sticky bit is on, and that FTPDNS exists in the search order.

If your system is not configured to display exit codes, look in the syslogd output for an FTP error message.

Connection terminated by the server after user enters password: If the server terminates a connection after the user enters a password, ensure that the FTP load modules (FTPD and FTPDNS) reside in the APF authorized data set and that all programs accessed by the FTP address space are APF authorized and marked as “controlled.” Additional symptoms include the following:

- The FTP daemon is running, but the FTP server address space abends.
- The FTP server trace is active with the ACC option and the last FTP trace entry reads:
RA0nnn pass: termid is ...

Connection terminated by the server after user enters any subcommand: If the server terminates a connection after the user enters a subcommand, either one or both of the following events may occur:

- FTP server address space shows an exit code of 0000.
- Last entry for the client session is RXnnnn Server thread terminates rc = -2. and the preceding entries indicate a “select” error due to a bad file descriptor.

These events indicate that the server inactive time limit has probably expired with no activity from the client. If this happens frequently, check the inactive time set for the server, increase it, if necessary, and recycle the FTP daemon.

Password validation fails; session continues: If password validation fails and the session continues, you may receive one of the following two replies from the FTP server:

- 530-error processing PASS command: <reason>
If you receive this message, do one or more of the following:
 - Ensure all libraries used by FTP are controlled and APF authorized.
 - Ensure FTP is authorized if you are using BPX.DAEMON.
 - Ensure that the FTP daemon has been started from a user ID running with superuser authority if the daemon has been started from the z/OS UNIX shell.
- 530-PASS command failed - getpwnam() error: <reason>
If you receive this message, ensure that the login user ID has an OMVS segment defined or that a default OMVS segment is established.

Anonymous logon fails: If an anonymous logon fails:

1. Ensure that you have specified ANONYMOUS as a start parameter or in FTP.DATA.
2. Check the setting of the ANONYMOUSLEVEL variable in FTP.DATA. If ANONYMOUSLEVEL is not explicitly set in FTP.DATA, its value will be equal to one.

If ANONYMOUS is set in FTP.DATA, and the STARTDIRECTORY is HFS, and ANONYMOUSLEVEL is two or three, verify that the required executable files are installed in the anonymous user’s root directory. SYSLOGD will have error messages if the required executable files are not installed in the anonymous user’s

home directory. For information on setting up the anonymous user's root directory, refer to the *z/OS Communications Server: IP Configuration Guide*.

If you did not specify a user ID on the ANONYMOUS start parameter or FTP.DATA statement, ensure that the user ID ANONYMO is defined to TSO and RACF and that it has a defined OMVS segment or that a default OMVS segment exists for your system. For information on the z/OS UNIX environment and its security considerations, refer to *z/OS UNIX System Services Planning*.

If you did specify a user ID on the ANONYMOUS start parameter or FTP.DATA statement, ensure that the specified user ID is defined to TSO and RACF and that the specified user ID has a defined OMVS segment or that a default OMVS segment exists for your system.

If ANONYMOUSLEVEL is two or three, verify that the STARTDIRECTORY value is compatible with the ANONYMOUSFILEACCESS value and that the FILETYPE value is compatible with the ANONYMOUSFILETYPESEQ, ANONYMOUSFILETYPEJES, and ANONYMOUSFILETYPESQL values. If ANONYMOUSLEVEL=3 and one of the following is coded: ANONYMOUS or ANONYMOUS/USERID/PASSWORD, the user will be prompted to enter an e-mail address as a password. Verify that the e-mail address entered by the user is consistent with the requirements of the EMAILADDRCHECK statement in FTP.DATA. If ANONYMOUS/USERID is coded, then the user must provide the password for USERID. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information on these FTP.DATA statements.

Wrong initial working directory: If the initial working directory is *userid* instead of an HFS directory, ensure that the STARTDIRECTORY HFS statement is specified in the FTP.DATA data set and that the \$HOME directory (defined or defaulted) exists for the login user ID.

Unable to open data connection message from server: If, after issuing a command such as RETR, STOR, or LIST, the client receives the message 425 Unable to open data connection from the server, check the FTP server trace for an error. (The trace option SOC should be active when you diagnose data connection errors. See the "Diagnosing FTP server problems with traces" on page 326 for information about starting the FTP server trace.) One possible trace entry is data_connect: bind() error...permission denied. If you see this trace entry, ensure that the FTP data connection port is reserved to OMVS in the PROFILE.TCPIP data set. Following is an example:

```
PORT
  20 TCP OMVS NOAUTOLOG ;FTP data port
  21 TCP FTPD1          ;FTP control port
```

Another possible trace entry is data_connect: seteuid(0) error...Permission denied. If you see this trace entry when the trace option ACC is active, ensure that FTP has BPX.DAEMON authority.

TLS problems: The FTP server and client provide a level of security using the Transport Layer Security (TLS) protocol. The FTP server and client use the services of System SSL as described in *z/OS System Secure Sockets Layer Programming*, SC24-5901. This manual describes how system SSL works and also contains a chapter on obtaining diagnostic information.

If you are experiencing problems with the TLS support, gather TLS trace information from FTP by activating security processing trace. You activate the trace before the

FTP server starts by adding the DEBUG SEC statement to the server's FTP.DATA file or after the server starts (and before client connection) by using the MODIFY operator command MODIFY jobname,DEBUG=(SEC).

One of the common problems with the TLS handshake is a mismatch in the ciphersuites supported by client and server. z/OS FTP supports the following ciphersuites (note that each has an associated number that is known to TLS):

```
SSL_NULL_MD5      ; 01
SSL_NULL_SHA      ; 02
SSL_RC4_MD5_EX    ; 03
SSL_RC4_MD5       ; 04
SSL_RC4_SHA       ; 05
SSL_RC2_MD5_EX    ; 06
SSL_DES_SHA       ; 09
SSL_3DES_SHA      ; 0A
```

The following is a portion of the FTP server trace for a successful TLS negotiation. In this example, the server of the FTP.DATA file was coded to accept only ciphersuites (cipherspecs) 01 and 02:

```
auth: entered with mechname TLS
ftpAuth: keyring = /u/user33/keyring/key.kdb
ftpAuth: stash   = /u/user33/keyring/key.sth
ftpAuth: environment_open()
ftpAuth: connect as a server
ftpAuth: environment_init()
ftpAuth: environment initialization complete
authClient: secure_socket_open()
authClient: cipherspecs = 0102
authClient: secure_socket_init()
tlsLevel: using TLSV1 with SSL_NULL_MD5 (01)
```

If the client were coded to not accept ciphersuites 01 and 02, the trace would look like this:

```
auth: entered with mechname TLS
ftpAuth: keyring = /u/user33/keyring/key.kdb
ftpAuth: stash   = /u/user33/keyring/key.sth
ftpAuth: environment_open()
ftpAuth: connect as a server
ftpAuth: environment_init()
ftpAuth: environment initialization complete
uthClient: secure_socket_open()
uthClient: cipherspecs = 0102
uthClient: secure_socket_init()
uthClient: init failed with rc = 402 (GSK_ERR_NO_CIPHERS)
ndSecureConn: entered
EYFT96I TLS handshake failed
```

Data transfer problems

This section describes various problems involving data transfer.

Load module transfer failures: This section describes failures when transferring MVS load modules.

If the MVS load module transfers but is not executable on the target system:

- Ensure that all hosts involved in the load module transfer are at the CS for OS/390 V2R10 level or higher.
 - For proxy transfers, both servers and the client must be CS for OS/390 V2R10 or higher.
- Ensure that the user did not attempt an operation which is not supported by load module transfer:

- Ensure that the user did not attempt to rename the load module on transfer.
- Ensure that the working directory on both the current and target systems is a load library of the correct type. An MVS load library for purposes of this support is a PDS with RECFM=U or a PDSE. Files may only be transferred between the same types of load libraries. This means that a PDS load library member must be transferred to another PDS, and a PDSE load library member must be transferred into another PDSE. The FTP client displays a terminal message EZA2841I Local directory might be a load library when a user changes local directory into a PDS or PDSE eligible for load module transfer support. The FTP server sends a 250-The working directory might be a load library reply to the client when a CWD command is processed which causes the server working directory to become a PDS or PDSE eligible for load module transfer support. If both the message and the reply are not seen when changing directories before a transfer, load module transfer processing will NOT be used to transfer any files between the two directories.
- Ensure that the load modules are transferred by member names only. The current working directory on both the target and destination systems must be the load library. Fully qualifying the member names is not permitted.
- Ensure that there are no problems with the IEBCOPY invocation. If an error is detected with an IEBCOPY invocation, the FTP server or client will furnish the IEBCOPY SYSPRINT output as messages to either the console (in the server's case) or the terminal session (in the client's case). The IEBCOPY SYSPRINT output can be displayed for both successful and unsuccessful transfers by specifying the FSC(2) debug option for the general trace for the FTP client and for the FTP server. At the client, enter debug fsc(2) before the transfer. See "Start tracing" on page 327 for information about how to set the trace for the server.

If the MVS load module fails to transfer:

- If Reload of the load library failed or Unload of the load library failed messages or replies are seen, then these messages indicate a problem with a call to the IEBCOPY system utility. Ensure that the IEBCOPY system utility is installed on the system and available to be called from application programs. If so, examine the FTP debug trace to determine if IEBCOPY was successfully invoked (see the "Diagnosing FTP server problems with traces" on page 326 for information about activating FTP syslog tracing.) (Some client environments, particularly REXX scripts running under the UNIX system services shell, are not fully authorized to call IEBCOPY). If IEBCOPY was successfully invoked, examine the IEBCOPY SYSPRINT output (described above) to see if IEBCOPY reported any errors.
- If allocation failure messages or replies are seen, then:
 - If the data set whose allocation failed is either the source or destination load library, ensure that no other process has allocated the load library for exclusive use.
 - If no data set name appears, or if the data set name ends in the characters XLMT, ensure that sufficient temporary DASD is available on the system. Load module transfer requires the use of sufficient temporary DASD to hold all data that could be transferred in one transfer command. Consider breaking up large mget or mput transfers into smaller groups to reduce the amount of required temporary DASD. If sufficient temporary DASD is not immediately available, then the setting of the AUTOMOUNT/NOAUTOMOUNT site option will regulate whether or not FTP attempts to mount additional temporary storage to complete a load module temporary file allocation request.

If the MVS load module transfer hangs, then most likely the system is waiting for temporary DASD to be mounted. If your system does not respond promptly to mount requests for temporary DASD, consider setting the NOAUTOMOUNT (LOC)SITE option on the hanging system, and breaking up large load module transfer mgets and mputs into smaller requests to reduce the requirement for temporary DASD.

Data set allocation fails: If data set allocation is failing (MKD, STOR/STOU, or APPE), check for the following:

- Issue the STAT command and check for problems with the variables that define data set characteristics (LRECL, RECFM, BLKSIZE, PRIMARY, SECONDARY, or DIRECTORY).
 - Do they all have a valid value defined?
 - If the variable is not listed in the STAT command output, no value is assigned to this variable. If no value is assigned to the variable, the value must be picked up from another source — either a model DCB or SMS. Does either the DCBDSN or DATACLASS (SMS) parameter have a valid value to provide a source for the missing variables?
 - If an SMS data class is specified, is SMS active at the server system? (Current SMS status is displayed as part of the output for the STAT command).
 - If an SMS data class is specified, do the data class definitions contain values for the missing variables?
 - Are both PRIMARY and SECONDARY either specified or not specified? If either PRIMARY or SECONDARY are specified, neither of the values will be picked up from an SMS data class. Both must be unspecified to pick up the value from SMS or both must be specified to override the SMS values.
 - If a model DCB is specified, are the characteristics of this data set valid for the data set being allocated?
- Issue the STAT command and check the PRIMARY, SECONDARY, and SPACETYPE values to determine how large the new data set will be. The VOLUME and UNIT value of the STAT command indicate where the data sets will be allocated. (If neither volume or unit is shown by the STAT command, data sets will be allocated on the system default SYSDA DASD.) Does the server system have sufficient space where the data sets will be allocated to allocate the data set? The SITE QDISK command provides information about the space available at the server system.
- Is the destination at the server site writable? Check with the operator at the server system to verify that the destination of the new data set is not write protected.

Data set allocation not picking up correct characteristics: If the data set is being allocated successfully, but the resulting data set does not have the expected data set characteristics, check for the following:

1. All values obtained from SITE variables.
 - Issue the STAT command to verify that the settings of all the SITE variables are correct. If any variables are missing from the STAT output, check for values specified for the DCBDSN or DATACLASS parameters. If a value is specified for the DCBDSN data set, go to Step 3 on page 320. If a value is specified for the DATACLASS parameter, go to Step 2 on page 320.
 - Check for variables overridden by a client. The VM and MVS FTP clients automatically issue SITE commands when doing a STOR, STOU, or APPE command. The values sent automatically by the client could be overriding

values set by specific SITE commands issued by the user. To prevent the VM or MVS client from automatically sending new SITE settings, issue the SENDSITE command at the client.

2. Values from SMS.

If the DATACLASS parameter has been specified, but the actual data set characteristics do not match the values in the specified SMS data class, issue the **STAT** command and check the information shown in the output from the STAT command for the following:

- Is SMS active at the server system? If SMS is not active, the SMS data class cannot be used to define the data set.
- Are values specified for any of the data set characteristic variables (LRECL, RECFM, BLKSIZE, PRIMARY, SECONDARY, RETPD, or DIRECTORY)? If these keywords are missing from the STAT output, no value is assigned to them and the data set characteristics should be picked up from the SMS data class. If, however, a value is present for any of these variables, the setting shown by the STAT command overrides any information in the SMS data class. To pick up the value from the data class, issue the SITE command with the keyword with no value (for example, SITE RECFM) to turn off the parameter setting.
- Is a value specified for the DCBDSN parameter? If a DCBDSN data set is specified, the values for LRECL, RECFM, BLKSIZE, and RETPD will be obtained from the model DCB data set and overrides any values in the SMS data class. Issue the **SITE DCBDSN** command to turn off the DCBDSN parameter setting.
- Check for variables overridden by a client. The VM and MVS FTP clients automatically issue SITE commands when doing a STOR, STOU, or APPE command. The values sent automatically by the client could be overriding values set by specific SITE commands issued by the user. To prevent the MVS or VM client from automatically sending new SITE settings, issue the **SENDSITE** command at the client.

3. Values from DCBDSN

If the DCBDSN parameter has been specified, but the actual data set characteristics do not match the characteristics of the specified data set, issue the **STAT** command and check the information shown in the output from the STAT command for the following.

- Are values specified for any of the data set characteristic variables (LRECL, RECFM, BLKSIZE, or RETPD)? If these keywords are missing from the STAT output, no value is assigned to them and the data set characteristics are picked up from the DCBDSN data set. If, however, a value is present for any of these variables, the setting shown by the STAT command overrides the values of the DCBDSN data set. To pick up the value from the DCBDSN data set, issue the **SITE** command with the keyword with no value (for example, SITE RECFM) to turn off the parameter setting.
- Check for variables overridden by a client. The VM and MVS FTP clients automatically issue SITE commands when doing a STOR, STOU, or APPE command. The values sent automatically by the client could be overriding values set by specific SITE commands issued by the user. To prevent the VM or MVS client from automatically sending new SITE settings, issue the **SENDSITE** command at the client.

MVS data set not found: If the server is not able to find the MVS data set, check for the following problems:

- Issue the **DIR** command to display the data set. Can the server find the data set to list it?
- Is the MVS data set at the server in the catalog? The server can only locate cataloged MVS data sets. Check the user level of access to the catalog. FTP servers at the z/OS V1R2 level and later display only the data sets to which the user has access.
- Was the *pathname* on the FTP command entered in single quotation marks? If not, the path name specified is appended to the end of the current working directory. Issue the **PWD** command to display the current working directory. If `current_working_directory.pathname` is not the correct name of the file, either change the current working directory with the **CWD** command or issue the correct data set name in single quotation marks as the *pathname*.

RETR, STOR, RNFR, RNTD, APPE, or DELE of data set fails: If RETR, STOR, RNFR, RNTD, APPE, or DELE for the data set fails, check for the following problems:

- Is the data set protected by a security system, such as RACF or HFS permission bits or a retention period?
- Is the data set being used at the server site by another program or user?
- Was the data set available to the system, or was it migrated or on an unmounted volume?
- Did the data set or member exist?
- For RETR or STOR commands, did a REST command immediately precede the RETR or STOR?

If so, the client is attempting to restart a file transfer. The server cannot detect certain REST argument errors until the RETR or STOR command is processed. If the trace options CMD and FSC are active, the server reply and server trace output provide insight into whether the REST command is implicated. Verify that the client and server have reestablished the original file transfer environment before attempting the restart.

The following problems apply to MVS data sets only:

- Did the specified path name follow MVS data set naming conventions?
- Was the requested data set a type of supported data set organization (PS, PDS, or PDS member) on a supported device type (DASD or tape)?
- Were the path name specifications consistent with the type of data set? For example, if a member was requested, was the data set a PDS?

REST fails: Use the STAT command to determine the current mode.

If mode is Block, report the problem to IBM. If the mode is Stream:

1. Verify that the server is configured for stream mode restarts. The server FEAT reply includes REST STREAM if the server is configured correctly.
2. Inspect the REST reply for more insight into the reason the server rejected the REST command. Refer to *z/OS Communications Server: IP and SNA Codes* for more information about FTP server replies.

Data transfer terminated: Check for the following problems:

- Is the data set at the server large enough to receive the data being sent? If not, use the **SITE** command to change the space allocation for new data sets.
- If storing a member of a PDS, is there room in the PDS for an additional member? Is there room in the PDS directory for another directory entry?

- Did the client send an ABOR command?
- Is the file type correct? For example, if filetype=SQL when it should be set to SEQ or JES, the host file being retrieved is assumed to be a SQL statement and FTP will attempt to connect to DB2 and submit the statement to DB2 for processing.

Client abends during RETR command data transfer: If the client abends while processing a RETR command, issue the **STAT** command and check the value of the checkpoint interval. If this value is greater than zero and data is being transferred in EBCDIC, either block mode or compressed mode, the server is sending checkpoint markers with the data being transferred. If the client being used does not support checkpoint/restart, this checkpoint information can cause unpredictable results, such as abends or data errors at the client. Change the setting of the checkpoint interval by issuing **SITE CHKPTINT=0**.

Data set disposition incorrect when transfer fails: If the data set disposition is incorrect when transfer fails, check for the following problems:

- Data sets cataloged instead of deleted.
 - Issue the STAT command and check the setting of the conditional disposition. If the STAT command output indicates New data sets will be cataloged if a store operation ends abnormally, the server will catalog new data sets even if the data transfer fails. To change this setting, issue the **SITE CONDDISP=DELETE** command.
 - Did the transfer fail because the FTP server was either abending or being terminated by a STOP or CANCEL command? If this is the case, the data set will be kept.
 - Is the client sending checkpoint information? If the data is being transferred in EBCDIC, either in block mode or compressed mode and the client has sent at least one checkpoint marker, the FTP server will keep the data set even if the conditional disposition is set to delete.
- Data sets deleted instead of cataloged.
 - Issue the STAT command and check the setting of the conditional disposition. If the STAT command output indicates New data sets will be deleted if a store operation ends abnormally, the server will delete new data sets if the data transfer fails. To change this setting, issue the **SITE CONDDISP=CATALOG** command.

Checkpoint markers do not appear to be sent: Issue the **STAT** command and check the settings for data transfer. Checkpoint information is only transferred in EBCDIC, with either block or compressed mode. The checkpoint interval must be greater than zero.

The sender of the data initiates the checkpoint information; therefore, checkpointing must be set on at the client for a STOR, STOU, or APPE, (for the MVS FTP client, this is done by issuing the **LOCSITE CHKPTINT=nn** command with a value larger than zero) and set on at the server (by issuing the **SITE CHKPTINT=nn** command with a value larger than zero) for a RETR.

LOADLIB directory information is not sent with module transfer: Issue the **STAT** command and check the settings for data transfer. Load module directory information is only sent for EBCDIC with a mode of either block or compressed.

The client you are using must support the SDIR command.

Server PDS member statistics not created or updated: ISPFSStats must be set to TRUE in order to create or update the statistics for the PDS Member when using PUT, MPUT, GET, MGET, or APPEND subcommands. For PUT, MPUT, or APPEND, make sure the server's ISPFSStats is set to TRUE. This can be determined by issuing the STAT command. If it is not set to TRUE, you can set it by using the SITE subcommand. For example, SITE ISPFSStats sets ISPFSStats to TRUE, and SITE NOISPFSStats sets ISPFSStats to FALSE.

If the PDS directory block is full, PDS member statistics will not be updated.

DB2 query support

This section describes how to use FTP server DB2 query support and how to diagnose SQL problems.

How to use FTP server SQL support: Before you can use the FTP server to submit queries to the DB2 subsystem, complete the following steps:

1. Start the DB2 subsystem.
2. BIND the DBRM called EZAFTPMQ. This must be done whenever the part EZAFTPMQ.CSQLMVS has been recompiled.
The DBRM must be bound into the plan named EZAFTPMQ, unless the keyword DB2PLAN was used in your FTP.DATA file to specify a different plan name.
If you are running multiple instances of the z/OS UNIX FTP server at different maintenance levels, you must use DB2PLAN in FTP.DATA for each server and specify unique plan names.
3. Grant execute privilege to the public for the plan created in the previous step.

To submit a query to DB2 through the FTP server, issue the following commands as necessary:

- **SITE FILETYPE=SQL**
- **SITE DB2=db2name** where *db2name* is the name of a DB2 subsystem at the host
- **RETR fname1 fname2** where *fname1* is a file at the host that contains a SQL SELECT statement

Symptoms of SQL problems: The following two tables show some symptoms and possible causes of SQL problems. Table 15 on page 324 shows problems that generate a reply beginning with 55x.

Table 15. SQL problems generating 55x replies (FTP Server)

Reply	Output file	Possible causes
Reply 551: Transfer aborted: SQL PREPARE/DESCRIBE failure	The output file contains the SQL code and error message returned by the DB2 subsystem.	<ul style="list-style-type: none"> A syntax error in the SQL statement in the host file. The time stamp in the load module is different from the BIND time stamp built from the DBRM (SQL code = -818). This occurs if a BIND was not done for the EZAFTPMQ DBRM that corresponds to the current load module, or if the server is not configured to use the correct DB2 plan name. If this is the problem, every SQL query submitted through the FTP server will fail.
Reply 551: Transfer aborted: unsupported SQL statement	No output is sent from the host.	The file type is SQL, but the host file being retrieved does not contain an SQL SELECT statement.
Reply 551: Transfer aborted: attempt to connect to <i>db2name</i> failed (<i>code</i>)	No output is sent from the host.	<ul style="list-style-type: none"> The site <i>db2name</i> specifies a nonexistent DB2 subsystem. The DB2 subsystem has not been started.
Reply 551: Transfer aborted: SQL not available. Attempt to open plan <planname> failed (DB2_reason_code).	No output is sent from the host.	<ul style="list-style-type: none"> BIND was not done for the specified plan. BIND was done for plan name other than EZAFTPMQ, but FTP.DATA does not contain a DBZPLAN statement to specify this planname. User does not have execute privilege for the DB2 plan being used by the FTP server.
Reply 550: SQL query not available. Can't load CAF routines.	No output is sent from the host.	The DSNLOAD library is not in the link list or the FTP server STEPLIB.
Note: For more information about the messages, refer to <i>z/OS Communications Server: IP and SNA Codes</i> .		

Table 16 shows other SQL problems.

Table 16. Other SQL problems (FTP Server)

Problem	Possible causes
Output file contains only the SQL SELECT statement.	<ul style="list-style-type: none"> The file type is SEQ, rather than SQL. If the file type is SEQ, a retrieve is done, but the host file is just sent back to the client. The query is not submitted to the DB2 subsystem. The SELECT is for a VIEW for which the user ID does not have DB2 select privilege. The DB2 subsystem returns an empty table.
Client closes the connection because server is not responding.	<p>The processing time needed by DB2 and FTP or both for the SQL query has exceeded the client's time limit for send or receive.</p> <p>An FTP server trace with the options FSC and SQL indicates the amount of SQL activity through FTP and the approximate time when each query was processed.</p>

JES support

This section describes the procedures to follow when JES output is not found and when remote job submission functions fail.

JES output not found (zero spool files): If the server is in JESINTERFACELEVEL=1 and FILETYPE=JES, and a job has been submitted but the output of the job cannot be found (that is, you get zero spool files from a DIR command), check the following:

- Is the job name correct? The job name must consist of the user ID followed by a single character.
- Was the job output spooled to the hold queue? The server will only be able to retrieve job output that is in the hold queue.

For example: If JESINTERFACELEVEL=2 then make sure the JESJOBNAME, JESSTATUS, and JESOWNER filters are set correctly with the STAT command.

If the server is in JESINTERFACELEVEL=2 and FILETYPE=JES, and a job has been submitted, but the output of the job cannot be found (that is, you get zero spool files from a DIR command), check the 125 reply message to verify that the JESOWNER, JESJOBNAME, and JESSTATUS filters are set to values that apply to your job. For example, if the JESJOBNAME=USER1* and the job submitted was USER2A, then use the SITE command to set the JES filter to the appropriate value to find the job requested. If the SITE command does not allow the end user to change the values of the three JES filters, refer to the *z/OS Communications Server: IP User's Guide and Commands* to determine if the proper Security Access Facility resources allow changing of the JES filters for the user.

Remote job submission functions fail: For problems with remote job submission, run the FTP JES trace to check for the following:

- Cannot allocate internal storage
- JES is not communicating
- JES unable to find output for the specified job ID
- Unable to acquire JES access
- Unknown return code from GET JES spool request

- JES unable to provide spool data set name now
- JES unable to get a job ID for a PUT or GET request
- JES PUT or GET aborted, job not found
- JES PUT or GET aborted, internal error
- JES PUT or GET aborted, timeout exceeded
- JES internal reader allocation failed
- JES user exit error

To trace the FTP JES activity, use the `DEBUG=(JES)` or `DUMP=(JES)` options of FTP syslog tracing. See “Diagnosing FTP server problems with traces” for information about activating FTP syslog tracing.

User exit routine is not invoked

If the user exit routine is not invoked, check the FTP trace in syslogd to see if the exit routine was loaded. FTCHKIP is loaded once by the FTP daemon during initialization. The remaining user exits (FTCHKPWD, FTCHKCMD, FTCHKJES, FTPOSTPR, and FTSPMFEX) are loaded in the FTP server address space for each client session.

For example, check for one of the following:

```
main: ret code from fndmembr() for FTCHKIP is: 4
main: user exit FTCHKIP not found. Bypassing fetch().
```

or

```
main: ret code from fndmembr() for FTCHKCMD is: 0
main: chkcmdexit successfully loaded
```

If you have user-written exit routines and the FTP server is not able to find them, ensure that the user-written exit routines exist in an APF-authorized partitioned data set which is in the search order.

FTP Messages and FTP trace entries

If messages and trace entries do not appear in the syslog output file, do one or more of the following:

- Ensure that syslogd is configured for daemon entries. The file `/etc/syslog.conf` must have an entry for `daemon.info` to get FTP messages or an entry for `daemon.debug` to get FTP messages and trace entries.
- Ensure that the files specified for daemon entries exist at the time that syslogd started. If not, you need to create the files and recycle syslogd.
- Ensure that the files specified for daemon entries have appropriate permission bits (for example, 666).
- Ensure that syslogd is active.

If messages and trace entries display on the system console, it means that syslogd cannot write to the files specified for daemon entries and that `/dev/console` is defined. Check that syslogd is configured correctly and that the files specified for daemon entries have appropriate permission bits (for example, 666).

Diagnosing FTP server problems with traces

Syslog tracing is available to aid in debugging z/OS UNIX FTP server problems. A number of methods are available to start, stop, or modify syslog daemon and server tracing:

- TRACE start option
- FTP.DATA DEBUG statement

- FTP.DATA DUMP statement
- MODIFY jobname,DUMP operator command
- MODIFY jobname,DEBUG operator command
- server SITE DEBUG command
- server SITE DUMP command

See “Start tracing during FTP initialization” and the *z/OS Communications Server: IP Configuration Reference*, for details about the TRACE start option and FTP.DATA statements. See “Controlling the FTP server traces with MODIFY operator command” on page 329 and the *z/OS Communications Server: IP System Administrator’s Commands* for details about the MODIFY operator command. See “Stop tracing” on page 328, “Tracing activity for one user” on page 328, and the *z/OS Communications Server: IP User’s Guide and Commands* for details about the SITE command.

Once a client has logged in to FTP, the client can issue SITE DEBUG or SITE DUMP commands to change tracing for that session only.

Where to find traces

The z/OS UNIX FTP server sends its trace entries to syslogd. The daemon.debug statement in /etc/syslog.conf specifies where syslogd writes FTP trace records.

```
#
# All ftp, rexecd, rshd
# debug messages (and above
# priority messages) go
# to server.debug.a
#
daemon.debug                /tmp/syslogd/server.debug.a
```

All z/OS UNIX FTP trace entries are written to the same HFS file.

Note: The TRACE parameter and MODIFY operator command options are issued to the FTP daemon and affect all client sessions that connect to the z/OS UNIX FTP server while tracing is active.

Refer to the *z/OS Communications Server: IP Configuration Guide* for more information about syslogd.

Start tracing

This section discusses two methods of starting the FTP server traces:

- During FTP initialization
- After FTP initialization

Start tracing during FTP initialization: You can use the TRACE start parameter, the TRACE statement, or the DEBUG and DUMP statements in FTP.DATA to begin tracing during FTP daemon initialization. This continues tracing for all FTP events for all FTP sessions. The trace data is routed to a file in your HFS through a definition in your syslogd configuration file (/etc/syslog.conf).

Tracing remains active until you issue a MODIFY operator command to end it. See “Controlling the FTP server traces with MODIFY operator command” on page 329.

Note: When you issue a MODIFY operator command to end tracing, tracing does not occur for any subsequent client sessions; however, tracing will continue for any sessions that were already connected.

Start tracing after FTP initialization: After initialization, you can enable tracing using an MVS MODIFY operator command to the FTP server listener process. See “Controlling the FTP server traces with MODIFY operator command” on page 329. Already established FTP connections are not affected by a MODIFY operator command. Only FTP connections that are established after the MODIFY operator command was issued will be subject to tracing.

If you have coded `DEBUGONSITE TRUE` and `DUMPONSITE TRUE` in the server’s `FTP.DATA` file, you can use the `SITE DEBUG` command and the `SITE DUMP` command, respectively, to change tracing after you log in to FTP. For example, if you want to add JES general tracing and JES extended tracing, you could enter the following:

```
SITE DEBUG=(JES) DUMP=(JES)
```

If you want to restrict the use of the `SITE` command to change the tracing and your installation has a security product that supports the `SERVAUTH` class, you can provide additional levels of access control. If the installation has activated the `SERVAUTH` class and provided a profile for the `SITE DEBUG` command, only users who have read access to the profile are allowed to use the `SITE DEBUG` command. The profile name is:

```
EZB.FTP.systemname.ftpdemonname.SITE.DEBUG
```

For example, if the procedure `FTPD` is used to start the server on system `MVS164`, the profile name is:

```
EZB.FTP.MVS164.FTPD1.SITE.DEBUG
```

The user’s `SITE DEBUG` command is rejected if the security product determines that the user does not have read access to the profile.

If the installation has activated the `SERVAUTH` class and provided a profile for the `SITE DUMP` command, only users who have read access to the profile are allowed to use the `SITE DUMP` command. The profile name is:

```
EZB.FTP.systemname.ftpdemonname.SITE.DUMP
```

For example, if the procedure `FTPD` is used to start the server on system `MVS164`, the profile name is:

```
EZB.FTP.MVS164.FTPD1.SITE.DUMP
```

The user’s `SITE DUMP` command is rejected if the security product determines that the user does not have read access to the profile.

Stop tracing

You stop global tracing using the `MODIFY` operator command. For example, suppose your FTP jobname is `FTPD1`. You would issue `F FTPD1,DEBUG=(NONE)` to stop global tracing. Already established FTP connections that were started with tracing enabled continue to produce trace output until the connections are terminated, but new connections start without tracing enabled.

If you have coded `DEBUGONSITE TRUE` in the server’s `FTP.DATA`, the FTP client can use a `SITE DEBUG=NONE` command to stop tracing. The `SITE` command affects only tracing for the current FTP session.

Tracing activity for one user

A filter can be specified so that the traces are active for certain clients that log on. Trace data can include both general and JES-related activity and includes data

such as parameter lists and storage areas. The filtering can be done by either IP address of the client or by user ID for the session.

A client could use the SITE DEBUG and SITE DUMP subcommands to write excessive debugging information to the syslog and effectively disable the syslog function. To prevent this, a RACF profile controls whether a client is allowed to use these parameters on the SITE subcommand. FTP uses the SERVAUTH resource class. The resource name will be EZB.FTP.<systemname>.<ftpdaemonname>.SITE.<tracename>. The lowest level will be tracename, which is either DEBUG or DUMP.

Controlling the FTP server traces with MODIFY operator command

The general trace for the FTP server can be started for all user IDs during initialization by specifying the TRACE parameter either as a start option in the FTP server start procedure, or by coding a DEBUG BAS statement in FTP.DATA.

The MODIFY operator command can be used after initialization to control the general and extended tracing for the FTP server. The command supports two parameters:

- DEBUG= for general tracing
- DUMP= for extended tracing

Each allows a filter to be specified so that the traces are active for certain clients that log in. The filtering can be done by either IP address of the client or by user ID for the session.

Note: The *jobname* is the name associated with the FTP daemon background job. It is documented in message EZYFT41I in the FTP services log. If you started the z/OS UNIX server using a proc named FTPD, the job name to use for the MODIFY operator command is probably FTPD1. As client sessions connect to the FTP server, the session process adopts the trace options currently active. These options remain in effect for the life of the client session process, regardless of subsequent MODIFY operator commands issued to the FTP daemon.

Controlling general tracing: To control the general trace, enter one of the following:

```
MODIFY jobname,DEBUG=(option_1,option_2,...,option_n,USERID(filter_name))
```

```
MODIFY jobname,DEBUG=(option_1,option_2,...,option_n,IPADDR(filter))
```

Where options are one of the following:

? Displays the status of the general traces.

Note: The status of the trace is displayed as a response to all uses of the operator MODIFY DEBUG command. The ? allows you to get the status without making a change.

ACC

The ACC trace shows the details of the login process.

ALL

This value is used to set all of the trace points.

Note: When the ALL parameter is processed, both the FSC and the SOC trace will be set to level 1.

BAS

This value is used to set a select group of traces that offer the best overall details without the copious output generated by certain trace options. Specifying this value is the same as the following:

```
MODIFY jobname,DEBUG=(CMD,INT,FSC,SOC)
```

CMD

The CMD trace shows each command and the parsing of the parameters for the command.

FLO

The FLO trace shows the flow of control within FTP. It is useful to show which services of FTP are used for an FTP request.

FSC(*n*)

The FSC trace shows details of the processing the file services commands APPE, STOR, STOU, RETR, DELE, RNFR, and RNTD. This trace can be very intense and therefore it allows you to specify levels of granularity for the trace points. The level 1 tracing that is specified by entering FSC or FSC(1) is the level normally used unless more data is requested by TCP/IP service group. The variable *n* can be a number in the range 1–8.

Level 1

This level covers the major steps of the file services processing. Included are:

- entry to a command processor
- determination of the type of file being processed
- choice of allocation method
- choice of open method
- choice of transfer routine
- recognition of end of file or data
- close and deallocation
- call for SMF processing

Level 2

This level provides more details for the major steps that are executed. These should be one-time events that enhance the information for the steps of level 1 tracing. An example would be some additional information about the allocation process.

Level 3

This level provides trace information of repetitive events that occur during the processing. For example, a trace for each full buffer (180K) of data that is received. Another example is a trace for each restart marker that is sent. The rate of repetition should be low enough that this level does not flood the trace.

Level 4

This level provides trace information of repetitive events that occur at a higher rate than those of level 3. For example, a trace for each time data must be moved to the top of a buffer before the next receive_data.

Level 5

This level provides trace information of repetitive events that occur at a higher rate than those of level 4. This is the most intense and will cover events such as the processing of each block of data.

Note: This level of tracing produces an extremely large amount of data and should not be used for large file transfers.

INT

The INT trace shows the details of the initialization and termination of the FTP session.

JES

The JES trace shows details of the processing for JES requests, such as when SITE FILETYPE=JES is in effect.

NONE

This value is used to turn off all of the traces.

PAR

The PAR trace shows details of the FTP command parser. It is useful for debugging problems in the handling of the command parameters.

SEC

The SEC trace shows the processing of security functions such as TLS and GSSAPI negotiations.

SOC(*n*)

The SOC trace shows details of the processing during the setup of the interface between the FTP application and the network as well as details of the actual amounts of data that is processed. This trace can be very intense and therefore it allows you to specify levels of granularity for the trace points. The level 1 tracing that is specified by entering SOC or SOC(1) is the level normally used unless more data is requested by the TCP/IP service group. The variable *n* can be a number from 1 to 8.

Level 1

This level covers the major steps of the socket services processing. Included will be the steps of initiating a connection and closing a connection.

Level 2

This level adds more detail for level 1 events. For example, it traces the three steps that occur when a data connection is closed.

Level 3

The events for this trace are the send() and recv() calls for the data connection.

SQL

The SQL trace shows details of the processing for SQL requests, such as when SITE FILETYPE=SQL is in effect.

UTL

The UTL trace shows the processing of utility functions such as CD and SITE.

USERID(*filter_name*)

This parameter is used to filter the trace for user IDs matching the *filter_name* pattern.

If the user ID matches the filter at the time the clients log in, then their tracing options will be set to the current value of the options. Otherwise, tracing options will not be set. Clients can use the SITE command to set their options after login if the initial ones are not appropriate. An example for the USERID filter is `MODIFY jobname,DEBUG=(CMD,USERID(USER3*))` which activates the CMD trace for a user whose ID starts with USER3.

IPADDR(*filter*)

This optional parameter is used to filter the trace for IP addresses matching the *filter* pattern.

If the IP address matches the filter at the time clients connect, then its tracing options will be set to the current value of the options. Otherwise, tracing options will not be set. Clients can use the SITE command to set their options after connect if the initial ones are not appropriate. An example of the IPADDR filter is `MODIFY jobname,DEBUG=(JES,IPADDR(9.67.113.57))`, which activates the JES trace for a client whose IP address is 9.67.113.57. Another example is `MODIFY jobname,DEBUG=(JES,IPADDR(FEDC:BA98:7654:3210:FEDC:BA98:7654:3210))`. This activates the JES trace for a client whose IP address is FEDC:BA98:7654:3210:FEDC:BA98:7654:3210.

If the filter is an IPv4 address, submasking can be indicated by using a slash followed by a dotted decimal submask. For example, `192.48.32/255.255.255.0` allows addresses from 192.48.32.00 to 192.48.32.255.

If the filter is an IPv6 address, network prefixing may be indicated by using a slash followed by a prefix length. For example, `FEDC:BA98::0/32` will allow all IP addresses from FEDC:BA98::0 to FEDC:BA98:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF.

The specification of the trace on the MODIFY operator command is *not* additive. That is, the trace setting is that of the last MODIFY operator command. For example:

```
MODIFY FTPDJG1,DEBUG=(NONE)
+EZYFT82I Active traces: NONE
MODIFY FTPDJG1,DEBUG=(CMD)
+EZYFT82I Active traces: CMD
MODIFY FTPDJG1,DEBUG=(FSC,USERID(USER33))
+EZYFT82I Active traces: FSC(1)
+EZYFT89I Userid filter: USER33
MODIFY FTPDJG1,DEBUG=(SOC)
+EZYFT82I Active traces: SOC(1)
```

The following are some guidelines to use for migrating from previous versions of the MODIFY operator command :

- `MODIFY jobname,TRACE`

This is still accepted and is equivalent to `MODIFY jobname,DEBUG=(BAS)`. The old response message EZY2704I is replaced by EZYFT82I.

- `MODIFY jobname,NOTRACE`

This is still accepted and is equivalent to `MODIFY jobname,DEBUG=(NONE)`. The old response message EZY2705I is replaced by EZYFT82I.

- `MODIFY jobname,JTRACE`

This is still accepted and is equivalent to `MODIFY jobname,DEBUG=(CMD,FSC,JES)`. The old response message EZY2710I is replaced by EZYFT82I.

- `MODIFY jobname,NOJTRACE`

This is still accepted and is equivalent to `MODIFY jobname,DEBUG=(NONE)`. The old response message EZY2711I is replaced by EZYFT82I.

- `MODIFY jobname,UTRACE=USER33`

This is rejected as an obsolete command. Its function can be replaced with the following pair of commands:

```
MODIFY jobname,DEBUG=(ALL,USERID(USER33))
MODIFY jobname,DUMP=(ALL,USERID(USER33))
```

- The use of the ALL parameter can produce an extensive amount of trace data and should not be specified on a routine basis.
- `MODIFY jobname,NOUTRACE`
This is rejected as an obsolete command. If complete tracing was activated as suggested in the previous step, then the tracing can be stopped as follows:
`MODIFY jobname,DEBUG=(NONE)`
`MODIFY jobname,DUMP=(NONE)`

Controlling extended tracing: To control the extended trace, enter one of the following:

`MODIFY jobname,DUMP=(option_1,option_2,...,option_n,USERID(filter_name))`

`MODIFY jobname,DUMP=(option_1,option_2,...,option_n,IPADDR(filter))`

Where options are one of the following:

id Specifies the ID number of a specific extended trace point that is to be activated in the FTP code. The ID number has a range of 1–99.

? Displays the status of the extended traces.

ALL

Activates all of the trace points.

NONE

Resets (turns off) all extended traces.

FSC

Activates all of the extended trace points in the file services code. The numbers activated are 20–49.

SOC

Activates all of the extended trace points in the network services code. The numbers activated are 50–59.

JES

Activates all of the extended trace points in the JES services code. The numbers activated are 60–69.

SQL

Activates all of the extended trace points in the SQL services code. The numbers activated are 70–79.

USERID(filter_name)

This parameter is used to filter the trace for user IDs matching the *filter_name* pattern.

If the user ID matches the filter at the time clients log in, then their tracing options will be set to the current value of the options. Otherwise, tracing options will not be set. Clients can use the SITE command to set their options after login if the initial ones are not appropriate. An example for the USERID filter is `MODIFY jobname,DEBUG=(21,USERID(USER33))`, which activates the dumpID 21 trace for a user if his user ID is USER33.

IPADDR(filter)

This parameter is used to filter the extended trace for IP addresses matching the *filter* pattern.

If the IP address matches the filter at the time clients connect, then its extended tracing options will be set to the current value of the options. Otherwise, tracing options will not be set. Clients can use the SITE command to set their options after connect if the initial ones are not appropriate. An example of the IPADDR

filter is `MODIFY jobname,DUMP=(JES,IPADDR(9.67.113.57))`, which activates the JES extended trace for a client whose IP address is 9.67.113.57. Another example is `MODIFY jobname,DUMP=(FSC,IPADDR(FEDC:BA98:7654:3210:FEDC:BA98:7654:3210))`. This activates all file services extended traces for a client whose IP address is `FEDC:BA98:7654:3210:FEDC:BA98:7654:3210`.

If the filter is an IPv4 address, submasking can be indicated by using a slash followed by a dotted decimal submask. For example, `192.48.32/255.255.255.0` allows addresses from `192.48.32.00` to `192.48.32.255`.

If the filter is an IPv6 address, network prefixing may be indicated by using a slash followed by a prefix length. For example, `FEDC:BA98::0/32` will allow all IP addresses from `FEDC:BA98::0` to `FEDC:BA98:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF`.

The specification of the trace on the `MODIFY` operator command is *not* additive. That is, the trace setting is that of the last `MODIFY` operator command. For example:

```
MODIFY FTPDJG1,DUMP=(NONE)
+EZYFT83I Active dumpIDs: NONE
MODIFY FTPDJG1,DUMP=(21)
+EZYFT83I Active dumpIDs: 21
MODIFY FTPDJG1,DUMP=(22)
+EZYFT83I Active dumpIDs: 22
```

The following are guidelines for migrating from the old parameters that were used with the `MODIFY` operator command.

- `MODIFY jobname,DUMP`
This format is rejected. `DUMP` requires at least one parameter (see above).
- `MODIFY jobname,NODUMP`
This is still accepted and is equivalent to `MODIFY jobname,DUMP=(NONE)`. The old response message `EZY2656I` is replaced by `EZYFT83I`.
- `MODIFY jobname,JDUMP`
This is rejected as an obsolete command with a suggestion to use the `DUMP` parameter. For example, use the command `MODIFY jobname,DUMP=(JES)`.
- `MODIFY jobname,NOJDUMP`
This is rejected as an obsolete command with a suggestion to use the `DUMP` parameter. For example, use the command `MODIFY jobname,DUMP=(NONE)`.

Logging FTP server activity

The z/OS FTP server provides a way to log standardized information for the following types of activity:

- Connections from the client end user to the server
- Authentication of the client/server session (for example, through the use of Transport Layer Security)
- Access to the FTP server through User ID/password verification
- Allocation of MVS data sets and HFS files
- Deallocation of MVS data sets and HFS files
- Data transfers
- JES job submissions
- SQL queries
- Abnormal end conditions

Logging can be enabled by setting the following server's FTP.DATA statements:

FTPLOGGING

ANONYMOUSFTPLOGGING

This information is recorded in the SYSLOGD file. The data has an identification field that allows correlation of all entries for a client/server session.

Refer to the *z/OS Communications Server: IP Configuration Reference* for the server's FTP.DATA configuration.

Documenting server problems

If the problem is not caused by any of the common errors described in this section, collect the following documentation before calling the IBM Support Center. Documentation is divided into two categories: essential and helpful but not essential.

- Essential:
 - Precise description of problem, including expected results and actual results
 - z/OS UNIX FTP server dump (for abends)
 - z/OS UNIX FTP server traces (see “Diagnosing FTP server problems with traces” on page 326 for information on collecting FTP server traces)
- Helpful:
 - FTP client output
 - FTP.DATA data set
 - TCPIP.DATA data set
 - PROFILE.TCPIP data set
 - ETC.SERVICES data set
 - The output from the STAT command issued to the server
 - If applicable, sample data to recreate the problem

FTP client

This section describes the procedures to follow when diagnosing problems with the FTP client.

Execution environments

The FTP client can run in any of the following environments:

- Interactive (under the TSO or the z/OS UNIX shell)
- Batch (under TSO only)
- REXX exec (under TSO)

When run interactively, you can redirect terminal I/O. When run under TSO, server responses and debug messages can be redirected to a file. For example, you can use the following command to redirect output from a TSO command line to a data set: **ftp 9.68.100.23 > 'USER27.FTPOUT'**. When run under the z/OS UNIX shell, both input and output can be redirected.

Note: When redirecting output under z/OS UNIX, nothing is displayed on the system console, not even command prompts, and it is difficult to know when input is requested. Consequently, use output redirection only when also using input redirection.

To redirect input from the file /user27/ftpinput and output to the file /user27/ftpout, issue the following command: **ftp 9.68.100.23 > /user27/ftpout < /user27/ftpinput**.

Setup

You can use an FTP.DATA data set to customize configuration parameters. You can use a SOCKS.CONFIG data set or file to instruct the client to connect to certain FTP servers through a SOCKS server. For information about the FTP.DATA data set and SOCKS configuration data set or file used by the FTP client, refer to *z/OS Communications Server: IP User's Guide and Commands*, *z/OS Communications Server: IP Configuration Guide*, and *z/OS Communications Server: IP Configuration Reference*.

Use the FTP client LOCSTAT command to display the name of the FTP.DATA file or the name of the SOCKS configuration data set or file that is being used.

The TCPIP.DATA configuration file provides information for the FTP client such as the high-level qualifier to be used for configuration data sets, and which DBCS translation tables can be used. For more information about the TCPIP.DATA configuration file, refer to the *z/OS Communications Server: IP Configuration Reference*. The z/OS UNIX search order for the file is used even if the FTP client is invoked under TSO.

Naming considerations

The FTP client can access both native MVS data sets and HFS files. For more information, see "Name considerations for z/OS UNIX FTP" on page 310.

Translation and data conversion support

This section describes translation and data conversion support for the FTP client.

Double-byte character set (DBCS) support

If the DBCS translate tables are not available, the client issues the following message after a valid command to establish a double-byte transfer type (for example, SJISKANKI, BIG5, or 'TYPE B n') is entered:

"EZA1865I Command not Supported. Translation Table not Loaded.

If this message displays, check the LOADDBCS statement in the TCPIP.DATA file. If the statement wraps to the next line, parameters on the continued line are ignored, and no error message is issued. If all parameters for the LOADDBCS statement do not fit on one line, use multiple LOADDBCS statements.

Check the precedence order for the TCPIP.DATA file to ensure that the file being used contains the LOADDBCS statement or statements. Be aware that the location of TCPIP.DATA statements can be influenced in multiple ways, for example, by a GlobalTCPIPData specification or the RESOLVER_CONFIG environment variable. Refer to the *z/OS Communications Server: IP Configuration Guide* for the TCPIP.DATA search order.

Single-byte character (SBCS) support

Data conversion occurs for single-byte data on the data connection when ENCODING=SBCS is in effect and the data type is ASCII. For more information, refer to the FTP.DATA statement ENCODING and the LOC SITE ENCODING subcommand in the *z/OS Communications Server: IP User's Guide and Commands*.

If you choose SBDATACONN as a statement in the FTP.DATA file or with the LOCSITE SBDATACONN subcommand, the FTP client builds a translation table using the codepages specified by SBDATACONN. If you receive the following message to the LOCSITE subcommand, start the trace with the DEBUG UTL option to determine which characters cannot be translated.

EZYFS08I

Some characters cannot be translated between *codepage_1* and *codepage_2*.

If none of the untranslatable characters appear in your data, your data transfers are not affected. If, however, one of the untranslatable characters does appear, your data transfer will fail and you will receive the following message:

EZA2930I

Transfer failed because data cannot be translated.

You can avoid the failure if you specify a substitution character to replace non-translatable characters. For more information on how to ask for character substitution, refer to SBSUB and SBSUBCHAR as FTP.DATA statements and as parameters on the LOCSITE subcommand in *z/OS Communications Server: IP User's Guide and Commands*. If substitution occurs during the transfer, you will receive the following message:

EZA2947I

One or more characters were substituted during the transfer.

When substitution occurs at the destination of a data transfer, a subsequent transfer of the resulting data will not produce an exact copy of the original. For example, if you get a file from the server and one or more characters are substituted, the untranslatable characters are overlaid with the substitution character. You cannot restore the original file by putting it to the server.

Multibyte character set (MBCS) support

Data conversion occurs for multibyte data on the data connection when ENCODING=MBCS is in effect and the data type is ASCII. For more information, refer to the FTP.DATA statement ENCODING and the LOCSITE ENCODING subcommand in the *z/OS Communications Server: IP User's Guide and Commands*.

If you choose ENCODING=MBCS, you must specify MBDATACONN with a statement in the FTP.DATA file or with the LOCSITE MBDATACONN subcommand to name the codepages for the multibyte data transfer. If you attempt an ASCII data transfer with ENCODING=MBCS and no MBDATACONN specified, you will receive the following message:

EZZ9793I

Multibyte encoding requested but codepages are not defined.

If the multibyte data that you transfer has codepoints that cannot be translated, the data transfer fails and you will receive the following message:

EZA2930I

Transfer failed because data cannot be translated

You can determine which bytes of the data cannot be translated by repeating the transfer with the DUMP 42 extended trace option active at the client.

File tagging support

When the server writes an HFS file, it may tag the file using the USS support for file tagging. In some cases you may experience conflicts when you try to read a file that has been tagged. A tagged HFS file has a file tag, an attribute that identifies the coded character set ID (ccsid) of the text data within the file. When a tagged file is read from the file system, the data is translated using the ccsid if SBDATACONN has specified a network transfer codepage to use with the file's codepage. A file may also be untagged or tagged binary.

ASCII file transfers

If you put data into an HFS file when the data type is ASCII, the file is tagged if you have used SBDATACONN to specify the codepage for the file system and for the network transfer [that is, you have specified SBDATACONN=(file_system_cp,network_transfer_cp)]. If the data conversion table is the FTP_STANDARD_TABLES or is specified using XLATE, the file is not tagged. The following example of a client session shows the effects of combining data type ASCII and SBDATACONN defined tables using codepages:

```
. . .
1 (01) Command:  ascii
(02) >>> TYPE A
(03) 200 Representation type is Ascii NonPrint
2 (04) Command:  site sbd=(ISO8859-1,ISO8859-1)
(05) >>> SITE sbd=(ISO8859-1,ISO8859-1)
(06) 200 Site command was accepted
3 (07) Command:  put afile
(08) >>> PORT 9,67,113,57,4,121
(09) 200 Port request OK.
(10) >>> STOR afile
(11) 125 Storing data set /u/user33/tagging2/afile
(12) 250 Transfer completed successfully.
(13) 200 bytes transferred in 0.070 seconds. Transfer rate 2.86 Kbytes/sec.
4 (14) Command:  site sbd=(IBM-1047,ISO8859-1)
(15) >>> SITE sbd=(IBM-1047,ISO8859-1)
(16) 200 Site command was accepted
5 (17) Command:  put afile
(18) >>> PORT 9,67,113,57,4,122
(19) 200 Port request OK.
(20) >>> STOR efile
(21) 125 Storing data set /u/user33/tagging2/efile
(22) 250 Transfer completed successfully.
(23) 200 bytes transferred in 0.005 seconds. Transfer rate 40.00 Kbytes/sec.
6 (24) Command:  site sbd=FTP_STANDARD_TABLES
(25) >>> SITE sbd=FTP_STANDARD_TABLES
(26) 200 Site command was accepted
7 (27) Command:  put afile
(28) >>> PORT 9,67,113,57,4,123
(29) 200 Port request OK.
(30) >>> STOR ufile
(31) 125 Storing data set /u/user33/tagging2/ufile
(32) 250 Transfer completed successfully.
(33) 200 bytes transferred in 0.005 seconds. Transfer rate 40.00 Kbytes/sec.
8 (34) Command:  ls -T
(35) >>> PORT 9,67,113,57,4,124
(36) 200 Port request OK.
(37) >>> NLST -T
(38) 125 List started OK
9 (39) t ISO8859-1  T=on  afile
10 (40) t IBM-1047  T=on  efile
11 (41) - untagged   T=off ufile
(42) 250 List completed successfully.
12 (43) Command:  get afile
(44) >>> PORT 9,67,113,57,4,125
(45) 200 Port request OK.
```

```

| (46) >>> RETR afile
| 13 (47) 125-Tagged ASCII file translated with current data connection translation table
| (48) 125 Sending data set /u/user33/tagging2/afile
| (49) 250 Transfer completed successfully.
| (50) 190 bytes transferred in 0.005 seconds. Transfer rate 38.00 Kbytes/sec.
| (51) Command: get efile
| (52) >>> PORT 9,67,113,57,4,126
| (53) 200 Port request OK.
| (54) >>> RETR efile
| 13 (55) 125-Tagged EBCDIC file translated with current data connection translation table
| (56) 125 Sending data set /u/user33/tagging2/efile
| (57) 250 Transfer completed successfully.
| (58) 200 bytes transferred in 0.005 seconds. Transfer rate 40.00 Kbytes/sec.
| (59) Command: get ufile
| (60) >>> PORT 9,67,113,57,4,127
| (61) 200 Port request OK.
| (62) >>> RETR ufile
| 14 (63) 125 Sending data set /u/user33/tagging2/ufile
| (64) 250 Transfer completed successfully.
| (65) 200 bytes transferred in 0.005 seconds. Transfer rate 40.00 Kbytes/sec.
| 15 (66) Command: site sbd=(IBM-1047,ISO8859-1)
| (67) >>> SITE sbd=(IBM-1047,ISO8859-1)
| (68) 200 Site command was accepted
| (69) Command: get afile
| (70) >>> PORT 9,67,113,57,4,128
| (71) 200 Port request OK.
| (72) >>> RETR afile
| 16 (73) 125-Tagged ASCII file translated with table built using file system cp=ISO8859-1,
|         network transfer cp=ISO8859-1
| (74) 125 Sending data set /u/user33/tagging2/afile
| (75) 250 Transfer completed successfully.
| (76) 190 bytes transferred in 0.005 seconds. Transfer rate 38.00 Kbytes/sec.
| (77) Command: get efile
| (78) >>> PORT 9,67,113,57,4,129
| (79) 200 Port request OK.
| (80) >>> RETR efile
| 17 (81) 125-Tagged EBCDIC file translated with table built using file system cp=IBM-1047,
|         network transfer cp=ISO8859-1
| (82) 125 Sending data set /u/user33/tagging2/efile
| (83) 250 Transfer completed successfully.
| (84) 200 bytes transferred in 0.005 seconds. Transfer rate 40.00 Kbytes/sec.
| 18 (85) Command: ebcdic
| (86) >>> TYPE E
| (87) 200 Representation type is Ebcdic NonPrint
| (88) Command: get afile
| (89) >>> PORT 9,67,113,57,4,142
| (90) 200 Port request OK.
| (91) >>> RETR afile
| 19 (92) 557 File contains ASCII data - enter TYPE A command before entering RETR command
| (93) Command: get efile
| (94) >>> PORT 9,67,113,57,4,143
| (95) 200 Port request OK.
| (96) >>> RETR efile
| 20 (97) 125 Sending data set /u/user33/tagging2/efile
| (98) 250 Transfer completed successfully.
| (99) 190 bytes transferred in 0.005 seconds. Transfer rate 38.00 Kbytes/sec.

```

Notes:

- 1** Change the data type to ASCII.
- 2** Site command requests a file system codepage ISO8859-1, an ASCII codepage.
- 3** Put a file and name it afile.
- 4** Site command requests a file system codepage IBM-1047, an EBCDIC codepage.

- 5** Put a file and name it efile.
- 6** Site command requests standard FTP translation tables.
- 7** Put a file and name it ufile.
- 8** The ls subcommand can be used to determine whether files in an hfs directory are tagged (that is, have a file tag). You use the -T option to request the file tagging information. When options are specified on the ls subcommand, name parameters cannot be specified.
- 9** afile is a tagged file. Its file system codepage is ISO8859-1. It is a Text file.
- 10** efile is a tagged file. Its file system codepage is IBM-1047. It is a Text file.
- 11** ufile is an untagged file. It is not a Text file.
- 12** Retrieve afile, which is a tagged file.
- 13** Client receives an indication that the tagged file is translated using the current tables because the current data connection tables were not specified with a network transfer codepage (see **6**).
- 14** Since this is an untagged file, no indication is needed about the tables used.
- 15** Specify translation tables with a file system codepage and a network transfer codepage.
- 16** The codepage of the tagged ASCII file is used with the network transfer codepage to translate the data in the file.
- 17** The codepage of the tagged EBCDIC file is used with the network transfer codepage to translate the data in the file.
- 18** Change the data type to EBCDIC.
- 19** The 557 reply informs the client that the data type must be ASCII when the file that is tagged as ASCII is retrieved.
- 20** The EBCDIC file is OK to send with data type EBCDIC since no translation occurs and the data is already EBCDIC.

Binary file transfers

If you put data into an HFS file when the data type is binary, the file is tagged as a binary file. The following example of a client session shows the effects of the binary file tagging:

```

1 (01) Command: binary
2 (02) >>> TYPE I
3 (03) 200 Representation type is Image
4 (04) Command: put a file
5 (05) >>> PORT 9,67,113,57,4,44
6 (06) 200 Port request OK.
7 (07) >>> STOR file
8 (08) 125 Storing data set /u/user33/newtag/file
9 (09) 250 Transfer completed successfully.
10 (10) 190 bytes transferred in 0.050 seconds. Transfer rate 3.80 Kbytes/sec.
11 (11) Command: ascii
12 (12) >>> TYPE A
13 (13) 200 Representation type is Ascii NonPrint
14 (14) Command: ls -T
15 (15) >>> PORT 9,67,113,57,4,45
16 (16) 200 Port request OK.
17 (17) >>> NLST -T
18 (18) 125 List started OK
19 (19) b binary      T=off file

```

```

| (20) 250 List completed successfully.
| (21) Command: get file
| (22) >>> PORT 9,67,113,57,4,46
| (23) 200 Port request OK.
| (24) >>> RETR file
| 4 (25) 557 File contains binary data - enter TYPE I command before entering RETR command
| (26) Command: binary
| (27) >>> TYPE I
| (28) 200 Representation type is Image
| (29) Command: get file
| (30) >>> PORT 9,67,113,57,4,47
| (31) 200 Port request OK.
| (32) >>> RETR file
| (33) 125 Sending data set /u/user33/newtag/file
| (34) 250 Transfer completed successfully.
| (35) 190 bytes transferred in 0.005 seconds. Transfer rate 38.00 Kbytes/sec.

```

Notes:

- 1 Request binary data type.
- 2 The ls subcommand can be used to determine whether files in an hfs directory are tagged. You use the -T option to request the file tagging information. When options are specified on the ls subcommand, name parameters cannot be specified.
- 3 The tagging information shows that the file is a binary file.
- 4 The 557 reply informs the client that the data type must be binary when the file is retrieved.

Common problems

This section describes some common problems with the FTP client.

Abends

If the client abends immediately after entering the FTP command and the following message is displayed, ensure that the local TSO user ID has an OMVS segment defined or that a default OMVS segment is established:

```

ftp
CEE5101C During initialization, the OpenEdition callable service
BPX1MSS failed. The system return code was 0000000156
, the reason code was 0B0C00FB . The application will be
terminated
IKJ56641I FTP ENDED DUE TO ERROR+
READY

```

Unknown host error message

The FTP client displays EZA1551I Unknown Host: <hostname> if it receives a negative response from the resolver. This occurs when the hostname specified on the FTP command cannot be resolved either by the name server or the local resolution file.

Note: The FTP client always uses the z/OS UNIX search order for TCPIP.DATA, even when FTP is invoked from TSO.

Use the host IP address instead of the hostname on the FTP command or see Chapter 15, “Diagnosing name server and dynamic domain name server (DDNS) problems” on page 441 for information on diagnosing name server problems.

Incorrect configuration values

Issue the **LOCSTAT** subcommand to determine the name of the file being used for your local site configuration parameters. If the file you want is not being used, start

the FTP client with the **-d** or **CTRACE** options to trace the client as it follows the search order for the FTP.DATA file. For more information about the search order used by the client, refer to *z/OS Communications Server: IP User's Guide and Commands*.

Determine whether your FTP.DATA file has sequence numbers. If it does, any statement with an optional parameter omitted will pick up the sequence number as the parameter value. For example, the BLKSIZE statement has an optional parameter *size*. If you specify the size, the sequence number is ignored. If you do not specify the size, the system assumes the sequence number is the size, causing an error.

Data transfer problems

Most of the data transfer problems that apply to a server also apply to a client. (See "Data transfer problems" on page 317.) In addition, a problem occurs when an FTP client is invoked under TSO and a TYPE U 2 or UCS2 command is invoked. When this happens, the following message is displayed:

```
EZA2749E Cannot establish conversion between <codeset>
          and UCS-2.
```

To transfer data encoded in UCS-2 during an FTP session, invoke the FTP command with the `_ICONV_UCS2_PREFIX` environment variable, specifying the prefix used for your runtime library. Following is an example:

```
FTP ENVAR("_ICONV_UCS2_PREFIX=CEE.OSVIR4") / <host_ip_addr> <port>
```

Server rejects PORT or EPRT command with 504 replies

Data transfer command sequences that use the PORT or EPRT command will fail when the server that receives the PORT or EPRT command is configured to reject all or certain PORT and EPRT commands. The reply code 504 indicates a problem of this nature.

For an ordinary transfer of data between client and server, the z/OS FTP client sends the PORT command to server when:

- the server does not support the EPSV command or the FTP session protocol is IPv4, and
- the client is not configured to be firewall friendly

You can correct this problem in one of these ways.

- Make the client firewall friendly. You can do this for the z/OS FTP client by coding `FWFRIENDLY TRUE` in the client's FTP.DATA, or by using a `LOCSITE FWFRIENDLY` subcommand before attempting the data transfer. The client sends EPSV or PASV to the server instead of PORT and the problem is avoided.
- Log in to the server using the server IPv6 address. (The server must have an IPv6 address.) The client uses EPSV instead of PORT and the problem is avoided.
- Change the server configuration so that it does not reject PORT or EPRT commands.
- Change the server so that it supports the EPSV command. The z/OS FTP server supports the EPSV command.

To change the client, see the *z/OS Communications Server: IP User's Guide and Commands* for information on the `FWFRIENDLY` statement and the `LOCSITE` subcommand.

If you used the proxy subcommand to start the transfer, you are transferring data between two servers instead of between client and server. For a transfer of data between two servers, the client must send PORT or EPRT to one of the servers, and PASV or EPSV to the other server. If the server receiving the PORT or EPRT command is configured to reject the PORT or EPRT command, the proxy transfer will fail with a 504 reply.

You can fix this problem in one of the following ways.

- Reverse the order in which you open the server connections. That is, if you opened a connection to ServerA and proxy opened a connection to ServerB, then open the connection to ServerB and proxy open the connection to ServerA. The client will then send PORT or EPRT to the other server during the proxy transfer. Provided the other server does not also reject PORT or EPRT, this will avoid the problem. Note however, that if the file you are transferring is a load module, changing the order in which you open server connections will not always cause the client to send PORT or EPRT to the other server.
- Transfer the file to a client, and then to the other server.
- Change the server so that it does not reject PORT and EPRT commands.

The following are z/OS server FTP.DATA statements that can be coded to reject PORT and EPRT commands:

PORTCOMMAND

Reject all PORT and EPRT commands.

PORTCOMMANDPORT

Reject PORT and EPRT commands whose port number argument is a well-known port number.

PORTCOMMANDIPADDR

Reject PORT and EPRT commands whose argument is an IP address that is different from the client's IP address.

Refer to *z/OS Communications Server: IP Configuration Guide* for more detail.

Client PDS member statistics not created or updated

ISPFStats must be set to TRUE in order to create or update the statistics for the PDS Member when using GET and MGET subcommands. When the PDS directory block is full, PDS member statistics are not updated. Use the LOCSTAT subcommand to verify that the client's ISPFStats setting is TRUE. You can change the ISPFStats value with the LOCSITE ISPFStats subcommand. Refer to *z/OS Communications Server: IP User's Guide and Commands* for information on using the LOCSITE subcommand.

Restarting file transfers

A valid restart of an interrupted file transfer depends on reestablishing the environment that existed at the time the file transfer failed. Environment includes:

- The current FTP.DATA statements
- The current SITE and LOCSITE settings
- The sequence of commands (such as Type, Mode, and Structure) that affect the way FTP transfers files
- The current translation tables in use on the data connection

All environment settings must be recreated before attempting to restart a file transfer. The following describe some possible problems that may be encountered.

Client rejects the RESTART subcommand:

- Verify that you have recreated the original file transfer environment
- Verify that your environment met all the restrictions for the restart subcommand
- Verify that checkpointing was active during the failed file transfer.
- Refer to *z/OS Communications Server: IP User's Guide and Commands* for information about restart subcommand restrictions and checkpointing a file transfer

Client rejects SRESTART subcommand:

- Verify that you have recreated the original file transfer environment
- Verify that the environment met the srestart subcommand restrictions
- Refer to the *z/OS Communications Server: IP User's Guide and Commands* for information about srestart subcommand restrictions. Unlike the restart subcommand, you do not need to activate check pointing, but you do need to enter the srestart parameters correctly

Client accepts SRESTART subcommand, but server rejects RESTART:

- Verify that the server supports stream mode restarts by issuing a FEAT command to the server. The FEAT reply will include the keyword REST_STREAM if the server supports stream mode restarts.
- Some FTP servers other than z/OS FTP servers reply to the FEAT command with REST_STREAM when they support stream mode restarts in one direction only, such as server to client file transfers. Contact the provider of the FTP server software to verify the server support stream restarts for the direction of the transfer you are attempting.
- Refer to *z/OS Communications Server: IP User's Guide and Commands* for information on the feature subcommand
- Verify that the server has recreated the environment extant during the failed file transfer.

Client ignores SOCKS configuration file

If you suspect that the client consistently ignores the SOCKS configuration file, use the LOCSTAT subcommand to display the name of the SOCKS configuration file.

- If no SOCKS configuration file name appears in the LOCSTAT output, the client is not configured correctly. Verify that a SOCKSCONFIGFILE statement is in FTP.DATA.
- Inspect the client syslog output for error messages relating to SOCKSCONFIGFILE in FTP.DATA. Use the client DEBUG INT statement to trace client initialization, and look for messages relating to the SOCKS configuration.

The FTP client references the SOCKSCONFIGFILE only when it is connecting to servers with IPv4 IP addresses; it is supposed to ignore the SOCKSCONFIGFILE when logging in to an FTP server with an IPv6 IP address. If you specify the FTP server by DNS name, that name might resolve to an IPv6 address rather than to an IPv4 address. You can use the LOCSTAT command to display the IP address used to log in to the server.

Client connects to wrong SOCKS server

If the client connects to a wrong SOCKS server; to a SOCKS server when it should not; or ignores SOCKS configuration file some of the time, use the LOCSTAT subcommand to display the name of the SOCKS configuration file.

- If the name displayed is not correct, correct the SOCKSCONFIGFILE statement in FTP.DATA.

- If the SOCKS configuration file name displayed by LOCSTAT is correct, inspect the contents of the SOCKS configuration file.

The client processes the statements in the order they are coded and applies the first statement that specifies the target FTP server. Check and arrange the statements as appropriate, or add a new statement specific to the FTP server at the beginning of the file.

Connection through SOCKS server to FTP server fails

A SOCKS connection involves a connection between the client and SOCKS server, and the SOCKS server and the target server. When a connection fails, try to isolate the point of failure.

- Can client connect to the SOCKS server host?
Use the client SOC(2) trace and the DUMP 85 trace during connection establishment, and inspect any messages to gain insight into whether the client was able to connect to the SOCKS server.
- Is the link between the client and the SOCKS server good?
Use ping to test the link.
- Is the SOCKS server active?
- Is the SOCKS server configured to reject the connection?
Contact the administrator of the SOCKS server for assistance.
- Is the link between the SOCKS server and the FTP server good?
Ask the administrators of the SOCKS server and the FTP server to verify the link.
- Is the FTP server active and accepting connections?
Contact the administrator of the FTP server. For the z/OS FTP server, activate the trace and check the syslog to determine whether the FTP server received a connection from the SOCKS server on behalf of the client.

DB2 query support

This section describes how to use the FTP client DB2 query support and how to diagnose SQL problems.

How to use FTP client SQL support

Before you can use the FTP client to submit queries to the DB2 subsystem, complete the following steps:

1. Start the DB2 subsystem.
2. BIND the DBRM called EZAFTPMQ. This must be done whenever the part EZAFTPMQ.CSQLMVS has been recompiled.
The DBRM must be bound into the plan named EZAFTPMQ, unless the keyword DB2PLAN was used in your FTP.DATA file to specify a different plan name.
3. Grant execute privilege to the public for the plan created in the previous step.

To use the FTP client to submit a query to DB2 and send the output to the FTP server, issue the following commands as necessary:

- **LOCSITE FILETYPE=SQL**
- **LOCSITE DB2=db2name** where *db2name* is the name of a DB2 subsystem at the local host
- **PUT fname1 fname2** where *fname1* is a local file that contains a SQL SELECT statement

Symptoms of SQL problems

The following two tables show some symptoms and possible causes of SQL problems. Table 17 shows problems that generate a reply beginning with 55x.

Table 17. SQL problems generating 55x replies (FTP Client)

Reply	Output file	Possible causes
EZA2570E: Transfer aborted: SQL PREPARE/DESCRIBE failure	The output file contains the SQL code and error message returned by the DB2 subsystem.	<ul style="list-style-type: none"> A syntax error in the SQL statement in the host file. The time stamp in the load module is different from the BIND time stamp built from the DBRM (SQL code = -818). This occurs if a BIND was not done for the EZAFTPMQ DBRM that corresponds to the current load module, or if the server is not configured to use the correct DB2 plan name. If this is the problem, every SQL query submitted through the FTP server will fail.
EZA2573E: Transfer aborted: unsupported SQL statement	No output is sent from the host.	The file type is SQL, but the host file being retrieved does not contain an SQL SELECT statement.
EZA2568E: Transfer aborted: attempt to connect to <i>db2name</i> failed (<i>code</i>)	No output is sent from the host.	<ul style="list-style-type: none"> The locsite <i>db2name</i> specifies a nonexistent DB2 subsystem. The DB2 subsystem has not been started.
EZA2569E: Transfer aborted: SQL not available. Attempt to open plan <planname> failed (DB2_reason_code).	No output is sent from the host	<ul style="list-style-type: none"> BIND was not done for the specified plan. BIND was done for plan name other than EZAFTPMQ, but FTP.DATA does not contain a DBZPLAN statement to specify this plan name. User does not have execute privilege for the DB2 plan being used by the FTP server.
EZA2740E: SQL query not available. Cannot load CAF routines.	No output is sent from the host.	The DSNLOAD library is not in the link list or the FTP server STEPLIB.
Note: For more information about these messages, refer to <i>z/OS Communications Server: IP and SNA Codes</i> .		

Table 18 shows other SQL problems.

Table 18. Other SQL problems (FTP Client)

Problem	Possible causes
Output file contains only the SQL SELECT statement.	<ul style="list-style-type: none">• The file type is SEQ, rather than SQL. If the file type is SEQ, a retrieve is done, but the local file is just sent to the server. The query is not submitted to the DB2 subsystem.• The SELECT is for a VIEW for which the user ID does not have DB2 select privilege. The DB2 subsystem returns an empty table.
Connection terminated.	<p>The processing time needed by DB2 or FTP or both for the SQL query has exceeded the server time limit for send or receive.</p> <p>If you are using the MVS FTP server and the server trace shows a select error due to a bad file descriptor, check the inactive time set for the server and, if necessary, increase the time.</p> <p>An FTP client trace indicates the amount of SQL activity through FTP and the approximate time when each query is processed.</p>

Diagnosing FTP client problems with tracing

You can activate tracing on startup with the -d command line option. Alternatively, you can activate tracing by toggling tracing on or off during an FTP session with the DEBUG command.

The DEBUG and DUMP subcommands activate the general and the extended levels of tracing. The general tracing shows key events in the processing of a subcommand (for example, the opening of a file) and the extended trace shows data areas that are used during processing. The extended trace produces large amounts of output and should be used at the direction of IBM service team. DEBUG 2 includes trace entries generated by level 1 and additionally dumps internal storage. The format of DEBUG allows multiple parameters to be specified on one subcommand. Refer to *z/OS Communications Server: IP User's Guide and Commands* for the syntax and parameters for the DEBUG and DUMP subcommands.

For example, the following sequence of subcommands would set traces:

```
DEBUG ACC SQL      *Activates the ACC and SQL traces
DEBUG BAS          *Activates the default traces
                   *CMD, INT, FSC, and SOC in addition
                   *to the two already set
DEBUG              *Resets all tracing
```

The parameter 2 is the equivalent of DEBUG BAS.

When running FTP interactively or from a REXX exec, all tracing goes to the terminal unless output is redirected. When running FTP from a TSO batch job, all tracing goes to SYSOUT.

If you experience problems using DD names to refer to files in FTP transfers, check the following:

1. Ensure that the user has properly allocated the DDNAME being referred to. The TSO command LISTALC STAT HIST can be helpful in debugging allocations. Also ensure that the allocations are proper (for example, if a file already exists, the disposition should not be NEW).
2. Ensure that DDNAMEs are only used to refer to local files. For example, get //DD:FTP01 FILEONE is not valid because it attempts to use a DDNAME to refer to a host file. If you try to use a DDNAME for a remote file name, the name will be sent to the remote host for processing as it is. If the remote host actually has a file named //DD:FTP01, then that file would be referred to, but most likely the remote host would reject it as a file name that is not valid.
3. To find attempts to access files by DDNAME, look for DD: in FTP trace output.

```
MF0573 seq_open_file: OSTN -> w,recfm=*,NOSEEK for dd:FTP02
MF0663 seq_open_fle: ddname FTP02 has filename USER1.CCPYXLMT
MF0669 seq_open_file: set DDNAME characteristics- recfm=90, lrecl=128, blksize=6144
```

Note: By using DDNAME support, the user is assuming responsibility for correctly allocating and deallocating the DDNAMEs being used.

Documenting FTP client problems

If the problem is not caused by any of the common errors described in this section, collect the following documentation before calling the IBM Software Support Center. Documentation is divided into two categories: essential and helpful but not essential.

- Essential:
 - Precise problem description, including client console, expected results, and actual results
 - Client trace output. You can use DEBUG ALL to capture all details possible.
- Helpful:
 - Output from the client LOCSTAT command
 - FTP.DATA data set
 - TCPIP.DATA data set
 - If appropriate, sample data to recreate the problem

Chapter 9. Diagnosing z/OS UNIX Telnet daemon problems

This chapter provides diagnostic information for z/OS UNIX Telnet daemon.

Common problems

The following list describes common problems that you may encounter during execution of the Telnet daemon.

- Diagnostic messages are not being printed to the appropriate file.
 - The diagnostic messages are printed out with the use of syslogd. Ensure that the syslogd is currently active by checking for /etc/syslog.pid.
 - If syslogd is active, ensure that the file where the output will be sent is currently allocated. Syslogd will not create the file; it expects it to exist. z/OS UNIX Telnet uses local1.debug for logging messages. Ensure that the syslog.conf file contains an entry for local1.debug or the *.* default file. Refer to the *z/OS Communications Server: IP Configuration Guide* for more detailed information about syslogd.
 - Ensure also that the specified file exists. Ensure that the permissions on the file are at a minimum "666".
 - Make sure you specify -t or -D all, or -t and -D all, as the z/OS UNIX Telnet options in /etc/inetd.conf.

- Use of the arrow keys.

The arrow keys are not functional in raw mode. This is AIX-like behavior, except that, in AIX, the arrow key produces peculiar characters such as ^--B on the screen to let the user know not to use arrows. Under rlogin, the cursor moves to where you would want it and correction is allowed, but the shell also treats these characters as part of the original command.

- The keyboard appears to be locked and the user cannot issue commands.

When executing UNIX-type clients (for example, AIX), if the -k option is specified for telnet in inetd.conf, telnet does not allow kludge linemode (see "Setting up the inetd configuration file" on page 463). UNIX-type clients require character-at-a-time mode to process correctly. If you remove the -k option from the parameters, then the software processes correctly.

If this does not work, run tracing -t D all. Look for **Ept** to determine what the exception conditions are for the **pty**. The number of bytes should equal four. Verify that the exception conditions identified are processed by the Telnet server. (Check EYZTE67I messages for more information; see Figure 27 on page 351.)

- EDC5157I An internal error has occurred, rsn=0b8802AF.

The "2AF" of the reason code signifies that the user did not have the proper authority to execute the command. This may result in either the user system having BPX.DAEMON authority set up in their environment, and the proper authorities have not been issued to the user, or the user does not have super user authority, which may be required to issue some of these commands.

Debug traces

Table 19 on page 350 describes options that relate to user-controlled trace information.

Table 19. Debug trace options

Option	Sub-Option	Description
-t		Internal tracing, intended to replace the DIAGNOSTICS compile option currently in place within the BSD code.
-D	authentication	Turns on authentication debugging code.
-D	encryption	Turns on encryption debugging code.
-D	options	Prints information about the negotiation of TELNET options.
-D	report	Prints the options information, plus some additional information about what processing is going on.
-D	netdata	Displays the data stream received by telnetd.
-D	ptydata	Displays the data stream written to the pty.
-D	all	Supports all of the options/report/ptydata/netdata/authentication/encryption options.

Debug trace flows (netdata and ptydata)

When issuing any of the following three trace commands within /etc/inetd.conf (-D ptydata, -D netdata, or -D all), you have the contents in both hexadecimal and ASCII, and the data being sent over the sockets or between the ttys in your syslogd file. If the user is having problems between the parent and the client, try the -D netdata option. If it is between the parent and the child, try the -D ptydata option. If both or either may apply, try the -D all option.

Each set of hexadecimal data is preceded by a three-letter tag. This tag represents the direction the data is flowing from. Figure 26 is a pictorial representation of this flow.

- Int—client to parent
- Ont—parent to client
- Ipt—child to parent
- Opt—parent to child

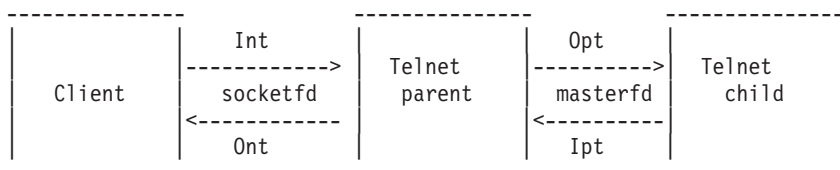


Figure 26. Trace between the Telnet client, parent, and child

The user types a command on the command line. It flows Int -> Opt. The child responds and the flow is Ipt -> Ont.

Debug trace examples (-t -D all)

Figure 27 on page 351 gives an example of the trace generated from -t -D all, generated from an AIX Telnet client. A trace explanation follows the figure.


```

1 EYZTE29I Starting new telnet session. catfd = 168443936
EZYTO05I Initial EBCDIC codepage = IBM-1047, ascii codepage = ISO8859-1
2 EYZTE05I Trace 1 Debug 3d keepalive 1 kludgelinemode 0
  hostinfo 1 Registered host 0 linemode 0 multi_proc 0
telnetd: doit(Second_pass=0)
3 EYZTE11I doit: host_name laph.raleigh.ibm.com
4 EYZTE11I doit: IP address 9.37.83.93
EYZTE11I doit: PORT      2504
EYZTE11I doit: host      MVSJ
>>>TELNETD: I support auth type 2 6
>>>TELNETD: I support auth type 2 2
>>>TELNETD: I support auth type 2 0
>>>TELNETD: I will support DES_CFB64
>>>TELNETD: I will support DES_OFB64
telnetd: getterminaltype() auth_level=0
state: send_do(option=37, init=1)
5 EYZTS04I STATE:send_do: send DO  AUTHENTICATION
6 EYZTU14I UTILITY: netwrite 3 chars.
7 EYZTU21I Ont: fffd25 ...
8 EYZTU03I UTILITY:ttloop read 33 chars.
9 EYZTU20I Int: fffb25fffd26fffb26fffd03fffb18fffb1ffffb .....
EYZTU20I Int: 20fffb21fffb22fffb27fffd05 .....
telrcv() encrypt_output=0
telrcv() decrypt_input =0
10 EYZTS05I STATE:willoption: receive WILL  AUTHENTICATION
>>>TELNETD: Sending type 2 6
>>>TELNETD: Sending type 2 2
>>>TELNETD: Sending type 2 0
utility: printsub(length=10)
11 EYZTU17I UTILITY: send suboption
  AUTHENTICATION
  SEND
  KERBEROS_V5
  CLIENT|MUTUAL|ENCRYPT
  KERBEROS_V5
  CLIENT|MUTUAL
  KERBEROS_V5
  CLIENT|ONE-WAY
12 EYZTS10I STATE:doooption: receive DO  ENCRYPT
13 EYZTS09I STATE:send_will: send WILL  ENCRYPT
14 EYZTS05I STATE:willoption: receive WILL  ENCRYPT
state: send_do(option=38, init=0)
15 EYZTS04I STATE:send_do: send DO  ENCRYPT
utility: printsub(length=6)
16 EYZTU17I UTILITY: send suboption
  ENCRYPT
  SUPPORT
  DES_CFB64
  DES_OFB64
17 EYZTS10I STATE:doooption: receive DO  SUPPRESS GO AHEAD
EYZTS09I STATE:send_will: send WILL  SUPPRESS GO AHEAD
18 EYZTS05I STATE:willoption: receive WILL  TERMINAL TYPE
state: send_do(option=24, init=0)
19 EYZTS04I STATE:send_do: send DO  TERMINAL TYPE
EYZTS05I STATE:willoption: receive WILL  NAWS
state: send_do(option=31, init=0)
EYZTS04I STATE:send_do: send DO  NAWS
EYZTS05I STATE:willoption: receive WILL  TSPEED
state: send_do(option=32, init=0)
EYZTS04I STATE:send_do: send DO  TSPEED
EYZTS05I STATE:willoption: receive WILL  LFLOW
state: send_do(option=33, init=0)

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 1 of 11)

```

EZYTS04I STATE:send_do: send DO    LFLOW
EZYTS05I STATE:willoption: receive WILL    LINEMODE
state: send_do(option=34, init=0)
EZYTS04I STATE:send_do: send DO    LINEMODE
EZYTS05I STATE:willoption: receive WILL    NEW-ENVIRON
state: send_do(option=39, init=0)
EZYTS04I STATE:send_do: send DO    NEW-ENVIRON
EZYTS10I STATE:doooption: receive DO    STATUS
EZYTS09I STATE:send_will: send WILL    STATUS
>>>TELNETD: in auth_wait.
EZYTU14I UTILITY: netwrite 50  chars.
EZYTU21I Ont: fffa25010206020200ffffb26fffd26fffa .....0.....
EZYTU21I Ont: 26010102ffff0fffb03fffd18fffd1ffffd20fffd .....0.....
EZYTU21I Ont: 21fffd22fffd27fffb05 .....
EZYTU03I UTILITY:ttloop read 512 chars.
EZYTU20I Int: fffa2503757365723532ffff0fffa25000206006e .....0.....>
EZYTU20I Int: 8201c6308201c2a003020105a10302010ea20703 b.F.b.B.....~....s..
EZYTU20I Int: 050020000000a38201126182010e3082010aa003 .....tb../b...b...
EZYTU20I Int: 020105a1101b0e4b52423339302e49424d2e434f ...~.....(|
EZYTU20I Int: 4da22b3029a003020103a12230201b04686f7374 (s.....~.....?..
EZYTU20I Int: 1b186d76736a2e7463702e72616c656967682e69 .._...../%......
EZYTU20I Int: 626d2e636f6da381c33081c0a003020101a10302 .._?_taC.a{.....~..
EZYTU20I Int: 0101a281b30481b01cbcb5a95fd2aa72297fae13 ..sa..a....z^K..."..
EZYTU20I Int: d12bd57b08d13133a485c8a4473c585733ded76e J.N#.J..ueHu.....P>
EZYTU20I Int: 711511dfae0a732e0f62329f2c1ec3bd19b35b53 .....C]..$.
EZYTU20I Int: 58e6ede82efb0c80d525a79d26708f5f78109a85 .W.Y....N.x....^...e
EZYTU20I Int: 54e17ca09ca7a245549229aeecd1a01125338a28 ..@..xs..k...J.....
EZYTU20I Int: bb58c3dd526136471c4c0f0688317ac3fefc7f83 ..C../...<..h.:C.."c
EZYTU20I Int: b808ea2bfb64f6ebb0b041cf5edd2f5e43a17f52 .....6.....;...~".
EZYTU20I Int: 13a299b7925d3e923df5ef18d690c523e1c35834 .sr.k).k.5..0.E..C..
EZYTU20I Int: 62213fdb0d206b894adec1e1437d9e696d6de8b3 .....i..A..'__Y.
EZYTU20I Int: 724c0ed1a48196308193a003020101a2818b0481 .<.Juao.a].....sa..a
EZYTU20I Int: 88f7048e7c7e2b092c0c5301f15d8ed82b92a60d h7..@=.....1).Q.kw.
EZYTU20I Int: 9c0524bb740e761ad609ffff09c2a13cbcd952ef .....0....B~..R..
EZYTU20I Int: 704a5a9426a6e2607cfe0d1a3fa9969ba8d20836 ..!m.wS-@.....zo.yK..
EZYTU20I Int: b8fdf73528d73abebdb7bbd7135d08e815896c62 ..7..P..].P.).Y.i%.
EZYTU20I Int: c06e44celdf73816969e95b77ab5b8d95d2618b9 {>...7..o.n:...R)...
EZYTU20I Int: 7b2abbe6c6d9adab0320a73aeb5e14f9373503d #..WFR[...x..V.|l.&.
EZYTU20I Int: 18a97c34a5698c5dc56364d871939fee0193fff0 .z@.v..)E..Q.l...l.0
EZYTU20I Int: fffa2605ffff0fffa260102ffff0fffa1f00780032 .....0.....0.....
EZYTU20I Int: fff0fffa2203010300036203 .0.....
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=10)
20 EZYTU17I UTILITY: receive suboption
AUTHENTICATION
NAME
u
s
e
r
5
2
>>>TELNETD: Got NAME [user52]
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=465)

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 2 of 11)

```

21 EZYTU17I UTILITY: receive suboption
AUTHENTICATION
IS
KERBEROS_V5
CLIENT|MUTUAL|ENCRYPT
AUTH 110 130 1 198 48 130 1 194 160 3 2 1 5 161 3 2 1 14 162 7 3 5 0 32 0 0 0
163 130 1 18 97 130 1 14 48 130 1 10 160 3 2 1 5 161 16 27 14 75 82 66 51 57 48
46 73 66 77 46 67 79 77 162 43 48 41 160 3 2 1 3 161 34 48 32 27 4 104 111 115
116 27 24 109 118 115 106 46 116 99 112 46 114 97 108 101 105 103 104 46 105 98
109 46 99 111 109 163 129 195 48 129 192 160 3 2 1 1 161 3 2 1 1 162 129 179 4
129 176 28 188 181 169 95 210 170 114 41 127 174 19 209 43 213 123 8 209 49 51
164 133 200 164 71 60 88 87 51 222
>>>REPLY:2: [3] (91)
6f 59 30 57 a0 03 02 01 05 a1 03 02 01 0f a2 4b
>>>REPLY:2: [2] (21)
75 73 65 72 35 32 40 4b 52 42 33 39 30 2e 49 42
22 telnetd: Kerberos5 identifies him as ``user52@KRB390.IBM.COM''
EZYTU14I UTILITY: netwrite 130 chars.
EZYTU21I Ont: fffa25020206036f593057a003020105a1030201 .....?.....~...
EZYTU21I Ont: 0fa24b3049a003020101a2420440117ac36284cd .s.....s.. :C.d.
EZYTU21I Ont: 65384025a9ee70511777fd91aa4c367edd20162f .. .z.....j.<.=....
EZYTU21I Ont: 736d40b6e61fae60d2c74c25aa610dcd10526eea _ .W...-KG<../....>.
EZYTU21I Ont: b096d7e7ed90a06f7a595cddb92369be741fff0 .oPX...?:.*..k..X..0
EZYTU21I Ont: fffa2502020602757365723532404b5242333930 .....
EZYTU21I Ont: 2e49424d2e434f4dfff0 ...(..|(.0
utility: printsub(length=4)
23 EZYTU17I UTILITY: receive suboption
ENCRYPT
REQUEST-START
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=5)
24 EZYTU17I UTILITY: receive suboption
ENCRYPT
SUPPORT
DES_OFB64
>>>TELNETD: He is supporting DES_OFB64 (2)
Creating new feed
utility: printsub(length=14)
25 EZYTU17I UTILITY: send suboption
ENCRYPT
IS
DES_OFB64
OFB64_IV 123 117 204 223 5 21 98 2
>>>TELNETD: (*ep->start)() returned 6
EZYTS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 16 chars.
EZYTU21I Ont: fffa260002017b75ccdf05156202fff0 .....#.....0
utility: printsub(length=7)
EZYTU17I UTILITY: receive suboption
NAWS
0 120 (120)
0 50 (50)
auth_wait: auth_context a080a30, validuser 3
auth_wait: auth_level 0
telnetd: authtel client name: user52 auth name: user52
state: send_do(option=38, init=0)
EZYTS04I STATE:send_do: send D0 ENCRYPT
state: send_do(option=24, init=1)
state: send_do(option=32, init=1)
state: send_do(option=35, init=1)
EZYTS04I STATE:send_do: send D0 XDISPLOC
state: send_do(option=39, init=1)
state: send_do(option=36, init=1)

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 3 of 11)

```

EZYTS04I STATE:send_do: send D0    OLD-ENVIRON
EZYTS09I STATE:send_will: send WILL    ECHO
EZYTU14I UTILITY: netwrite 12  chars.
EZYTU21I Ont: fffd26fffd23fffd24fffb01 .....
EZYTU03I UTILITY:ttloop read 47 chars.
EZYTU20I Int: 04020f05030007621c08020409421a0a02080b02 .....
EZYTU20I Int: 150c02170d02120e02160f021110021311020012 .....
EZYTU20I Int: 0200fff0fffd03 ...0...
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=52)
EZYTU17I UTILITY: receive suboption
LINEMODE
SLC
SYNCH
DEFAULT
0;
IP
VARIABLE
|FLUSHIN|FLUSHOUT
3;
AO
VARIABLE
15;
AYT
DEFAULT
0;
ABORT
VARIABLE
|FLUSHIN|FLUSHOUT
28;
EOF
VARIABLE
4;
SUSP
VARIABLE
|FLUSHIN
26;
EC
VARIABLE
8;
EL
VARIABLE
21;
EW
VARIABLE
23;
RP
VARIABLE
18;
LNEXT
VARIABLE
22;

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 4 of 11)

```

XON
VARIABLE
17;
XOFF
VARIABLE
19;
FORW1
VARIABLE
0;
FORW2
VARIABLE
0;
EZYTS10I STATE:doooption: receive DO SUPPRESS GO AHEAD
EZYTU03I UTILITY:ttloop read 16 chars.
EZYTU20I Int: fffa26000201094321d752693162fff0 .....P.....0
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=14)
26 EZYTU17I UTILITY: receive suboption
ENCRYPT
IS
DES_OFB64
OFB64_IV 9 67 33 215 82 105 49 98
CFB64: initial vector received
Initializing Decrypt stream
utility: printsub(length=6)
27 EZYTU17I UTILITY: send suboption
ENCRYPT
REPLY
DES_OFB64
OFB64_IV_OK
(*ep->is)(a09abc3, 9) returned MORE_TO_DO (7)
EZYTU14I UTILITY: netwrite 8 chars.
EZYTU21I Ont: fffa26020202fff0 .....0
EZYTU03I UTILITY:ttloop read 17 chars.
EZYTU20I Int: fffa26020202fff0fffc23fffc24fffd01 .....0.....
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=6)
28 EZYTU17I UTILITY: receive suboption
ENCRYPT
REPLY
DES_OFB64
OFB64_IV_OK
utility: printsub(length=5)
29 EZYTU17I UTILITY: send suboption
ENCRYPT
ENC_KEYID
0
(*ep->reply)(a09abc3, 1) returned MORE_TO_DO (4)
>>>TELNETD: encrypt_reply returned 4
30 EZYTS08I STATE:wontoption: receive WON'T XDISPLOC
EZYTS08I STATE:wontoption: receive WON'T OLD-ENVIRON
EZYTS10I STATE:doooption: receive DO ECHO
>>>TELNETD: in encrypt_wait
EZYTU14I UTILITY: netwrite 7 chars.
EZYTU21I Ont: fffa260700fff0 .....0
EZYTU03I UTILITY:ttloop read 7 chars.
EZYTU20I Int: fffa260700fff0 .....0
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=5)

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 5 of 11)

```

31 EZYTU17I UTILITY: receive suboption
ENCRYPT
ENC_KEYID
0
utility: printsub(length=5)
29 EZYTU17I UTILITY: send suboption
ENCRYPT
DEC_KEYID
0
EZYTU14I UTILITY: netwrite 7 chars.
EZYTU21I Ont: fffa260800fff0 .....0
EZYTU03I UTILITY: ttloop read 14 chars.
EZYTU20I Int: fffa260800fff0fffa260300fff0 .....0.....0
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=5)
31 EZYTU17I UTILITY: receive suboption
ENCRYPT
DEC_KEYID
0
>>>TELNETD: Encrypt start: initial negotiation in progress (0) DES_OFB64
utility: printsub(length=5)
32 EZYTU17I UTILITY: send suboption
ENCRYPT
START
>>>TELNETD: Started to encrypt output with type DES_OFB64
EZYTU14I UTILITY: netwrite 7 chars.
EZYTU21I Ont: fffa260300fff0 .....0
utility: printsub(length=5)
33 EZYTU17I UTILITY: receive suboption
ENCRYPT
START
>>>TELNETD: Start to decrypt input with type DES_OFB64
utility: printsub(length=5)
EZYTU17I UTILITY: send suboption
ENCRYPT
REQUEST-START
>>>TELNETD: Request input to be encrypted
>>>TELNETD: Encrypt start: initial negotiation in progress (0) DES_OFB64
utility: printsub(length=5)
EZYTU17I UTILITY: send suboption
ENCRYPT
START
>>>TELNETD: Started to encrypt output with type DES_OFB64
telnetd: getterminaltype() auth_negotiated=1
utility: printsub(length=4)
34 EZYTU17I UTILITY: send suboption
TERMINAL-TYPE
SEND
EZYTU14I UTILITY: netwrite 32 chars.
EZYTU21I Ont: b306079a3675d15d45511c7172c0579b93f4b0ac .....J).....{..14..
EZYTU21I Ont: afdc1e09ae1fe760ed4d59b4 .....X-.(..
EZYTU03I UTILITY: ttloop read 51 chars.
EZYTU20I Int: b306079c3675d15d4551f48bb00984d8ac37f6c .....J).....q(.C"%
EZYTU20I Int: afd6c6f276ef18cfa609f445b8b52a92fb4c2a25 .OF2....w.4....k.<..
EZYTU20I Int: ca539aa8f2401960713bc5 ...y2 .-...E
telrcv() encrypt_output=0xA00C848
telrcv() decrypt_input =0xA00C6C0
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=5)
EZYTU17I UTILITY: receive suboption

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 6 of 11)


```

ENCRYPT
START
>>>TELNETD: Start to decrypt input with type DES_0FB64
EZYTS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=13)
EZYTU17I UTILITY: receive suboption
    TERMINAL-SPEED
    IS 9600,9600
EZYTS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=16)
EZYTU17I UTILITY: receive suboption
    NEW-ENVIRON
    IS
    VAR
    U
    S
    E
    R
    VALUE
    u
    s
    e
    r
    5
    2
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=9)
EZYTU17I UTILITY: receive suboption
    TERMINAL-TYPE
    IS XTERM
EZYTE10I terminaltypeok: call tgetent (buf, XTERM)
EZYTE51W terminaltypeok: Tgetent failure EDC5129I No such file or directory.
rsn = 0594003D
35 EZYTE10I terminaltypeok: call tgetent (buf, xterm)
telnetd: getterminaltype() return 3
EZYTO01I Int: 75 .
EZYTO02I Ont: 49 .
EZYTO01I Int: 73 .
EZYTO02I Ont: b7 .
EZYTO01I Int: 65 .
EZYTO02I Ont: 50 &
EZYTO01I Int: 72 .
EZYTO02I Ont: 77 .
EZYTO01I Int: 35 .
EZYTO02I Ont: 91 j
EZYTO01I Int: 32 .
EZYTO02I Ont: f6 6
EZYTO01I Int: 0d .
EZYTO02I Ont: e1 .
EZYTE59I read_pw: Character ignored 0
36 EZYT004I lusername = user52
telnetd: krb name: user52, user: user52
EZYTE22I herald()
37 EZYTE26E herald: stat error EDC5129I No such file or directory.
rsn = 053B006C
EZYTE16I uid = 52, gid = 5
telnsave: mallocTelnetSave() rc=0
telnetd: doit() subcount=96
telnetd: doit() execvp()
EZYTU34I id 30002 pri 3 call catopen(tnmsgs.cat,0) code 81 reason 053B006C
h_errno N/A
telnetd: main() -y getsubopt(tSave=2137884232)
telnsave: freeTelnetSave() rc=0

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 7 of 11)


```

EZYTO11I DInt: fffa1f00780032fff0ffa220301030003620304 .....0.....
EZYTO11I DInt: 020f05030007621c08020409421a0a02080b0215 .....
EZYTO11I DInt: 0c02170d02120e02160f02111002131102001202 .....
EZYTO11I DInt: 00fff0ffa2000393630302c39363030fff0ffa ..0.....0..
EZYTO11I DInt: 2700005553455201757365723532fff0 .....0
telnetd: doit(Second_pass=1)
EZYTY02I GETPTY: open of /dev/ptyp EDC5114I Resource busy. rsn = 020A0155
EZYTY02I GETPTY: open of /dev/ptyp EDC5114I Resource busy. rsn = 020A0155
EZYTY05I GETPTY: slave fd = 9 , masterfd = 8
telnetd: doit() deferred_processing=1
[38] EZYTS15I STATE:doooption:deferred receive D0 ECHO
EZYTO09I options(1) = 3 .
EZYTS15I STATE:doooption:deferred receive D0 SUPPRESS GO AHEAD
EZYTO09I options(3) = 3 .
EZYTS15I STATE:doooption:deferred receive D0 STATUS
EZYTO09I options(5) = 3 .
[38] EZYTS16I STATE:willoption:deferred receive WILL TERMINAL TYPE
EZYTO09I options(24) = 12 .
EZYTS16I STATE:willoption:deferred receive WILL NAWS
EZYTO09I options(31) = 12 .
EZYTS16I STATE:willoption:deferred receive WILL TSPEED
EZYTO09I options(32) = 12 .
EZYTS16I STATE:willoption:deferred receive WILL LFLOW
EZYTO09I options(33) = 12 .
EZYTS16I STATE:willoption:deferred receive WILL LINEMODE
EZYTU14I UTILITY: netwrite 13 chars.
EZYTU21I Ont: b7c2d637e5d127b4aeced4cbad .B0.VJ....M.[
EZYTO09I options(34) = 12 .
EZYTS16I STATE:willoption:deferred receive WILL AUTHENTICATION
EZYTO09I options(37) = 12 .
EZYTS15I STATE:doooption:deferred receive D0 ENCRYPT
EZYTO09I options(38) = 15 .
EZYTS16I STATE:willoption:deferred receive WILL NEW-ENVIRON
EZYTO09I options(39) = 12 .
telrcv() encrypt_output=0xA00C848
telrcv() decrypt_input =0xA00C6C0
EZYTS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=7)
EZYTU17I UTILITY: receive suboption
NAWS
0 120 (120)
0 50 (50)
EZYTS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=52)

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 8 of 11)

```

EZYTU17I UTILITY: receive suboption
LINEMODE
SLC
SYNCH
DEFAULT
0;
IP
VARIABLE
|FLUSHIN|FLUSHOUT
3;
AO
VARIABLE
15;
AYT
DEFAULT
0;
ABORT
VARIABLE
|FLUSHIN|FLUSHOUT
28;
EOF
VARIABLE
4;
SUSP
VARIABLE
|FLUSHIN
26;
EC
VARIABLE
8;
EL
VARIABLE
21;
EW
VARIABLE
23;
RP
VARIABLE
18;
LNEXT
VARIABLE
22;
XON
VARIABLE
17;
XOFF
VARIABLE
19;
FORW1
VARIABLE
0;
FORW2
VARIABLE
0;
EZYTS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=13)
EZYTU17I UTILITY: receive suboption
TERMINAL-SPEED
IS 9600,9600
EZYTS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=16)

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 9 of 11)

```

EZYTU17I UTILITY: receive suboption
NEW-ENVIRON
IS
VAR
U
S
E
R
VALUE
u
s
e
r
5
2
telnetd: doit() deferred_processing=0
state: send_do(option=31, init=1)
state: send_do(option=33, init=1)
telrcv() encrypt_output=0xA00C848
telrcv() decrypt_input =0xA00C6C0
state: send_do(option=0, init=1)
EZYTS04I STATE:send_do: send D0 BINARY
EZYTS07I STATE:send_dont: send DON'T LINEMODE
EZYTU14I UTILITY: netwrite 66 chars.
EZYTU21I Ont: 2e925cb2dbf210c689e053fbad994f522421dbb7 .k*..2.Fi\..[r].....
EZYTU21I Ont: 062ef6d290bf59f7e40600bc4c43f4eef139b405 ..6K...7U...<.4.1...
EZYTU21I Ont: a59b84b3fc185a609644499e56c69fe7790b6e7e v.d...!-o....F.X`.>=
EZYTU21I Ont: 8cdd2ede8d42 .....
utility: printsub(length=52)
EZYTU17I UTILITY: send suboption
LINEMODE
SLC
SYNCH
NOSUPPORT
0;
IP
VARIABLE
|ACK|FLUSHIN|FLUSHOUT
3;
AO
VARIABLE
|ACK
15;
AYT
NOSUPPORT
0;
ABORT
VARIABLE
|ACK|FLUSHIN|FLUSHOUT
28;
EOF
VARIABLE
|ACK
4;
SUSP
VARIABLE

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 10 of 11)

```

|ACK|FLUSHIN
26;
EC
VARIABLE
|ACK
8;
EL
VARIABLE
|ACK
21;
EW
VARIABLE
|ACK
23;
RP
VARIABLE
|ACK
18;
LNEXT
VARIABLE
|ACK
22;
XON
VARIABLE
|ACK
17;
XOFF
VARIABLE
|ACK
19;
FORW1
VARIABLE
|ACK
0;
FORW2
NOSUPPORT
0;
EZYTU14I UTILITY: netwrite 0 chars.
EZYTE66I PROTOCOL: lmodetype=4, linemode=0, uselinemode=0
39 EZYTY08I argv_fsum(0) = fomtlinp
EZYTY08I argv_fsum(1) = *40urhrEa)R0,H/h
EZYTY08I argv_fsum(2) =
EZYTY08I argv_fsum(3) = 0
EZYTY08I argv_fsum(4) = 8
EZYTY08I argv_fsum(5) = 9
EZYTY08I argv_fsum(6) = 0
EZYTY08I argv_fsum(7) = 0
EZYTY08I argv_fsum(8) = 6
EZYTY08I argv_fsum(9) = 80
EZYTY08I argv_fsum(10) = laph.raleigh.ibm.com
EZYTY08I argv_fsum(11) = xterm
EZYTY08I argv_fsum(12) =
EZYTY08I argv_fsum(13) =
EZYTY08I argv_fsum(14) =
EZYTY08I argv_fsum(15) =
EZYTY08I argv_fsum(16) = 1
EZYTY08I inherit flag = 40000000
EZYTY09I login_tty: spawnp fsumoclp 33
40 EZYTE67I S(nfd):socketfd..ibits=00000000 obits=00000000 ebits=00000000
S(nfd) pty..ibits=00000000 obits=00000000 ebits=00000100
41 EZYTE68I Ept: #bytes = 4 pkcontrol(cnt1) 1003
EZYTE69I PROTOCOL: cnt1 = 1003
EZYTE65I PROTOCOL: send IAC Data Mark. DMARK

```

Figure 27. z/OS UNIX Telnet trace using -t -D all (Part 11 of 11)

Following are short descriptions of the numbered items in the trace:

- 1** EZYTE29I indicates the start of a new z/OS UNIX Telnet client session.
- 2** EZYTE05I indicates what options were specified in /etc/inetd.conf for z/OS UNIX Telnet.
- 3** EZYTE11I indicates the resolved host name (from the client).
- 4** EZYTE11I shows the IP address of the z/OS UNIX Telnet client.
- 5** EZYTS04I indicates otelnetd is willing to send and receive authentication information.
- 6** EZYTU14I traces netwrites (writes to the client terminal).
- 7** EZYTU21I traces data from parent to client; that is, z/OS UNIX Telnet to the client terminal.
- 8** EZYTU03I indicates the number of bytes read from the client by z/OS UNIX Telnet.
- 9** EZYTU20I traces data from the client to the parent (z/OS UNIX Telnet server).
- 10** EZYTS05I indicates the client is willing to send and receive authentication information.
- 11** EZYTU17I shows otelnetd requesting that the client send authentication information for Kerberos Version 5.
- 12** EZYTS10I indicates the client is willing to receive encrypted data.
- 13** EZYTS09I indicates otelnetd is willing to send encrypted data.
- 14** EZYTS05I indicates the client is willing to send encrypted data.
- 15** EZYTS04I indicates otelnetd is willing to receive encrypted data.
- 16** EZYTU17I shows which types of encryption otelnetd will support when receiving data.
- 17** EZYTS10I shows the terminal option negotiation the client has sent/received.
- 18** EZYTS05I shows the terminal option negotiation the client has sent/received.
- 19** EZYTS04I indicates the terminal negotiation options sent to the client by the z/OS UNIX Telnet server.
- 20** EZYTU17I shows the account name on otelnetd that the client wishes to be authorized to use.
- 21** EZYTU17I shows the client authentication information for Kerberos Version 5.
- 22** Shows the Kerberos Version 5 principal of the user logging in.
- 23** EZYTU17I shows the client requesting that otelnetd enable encryption as soon as the initialization is completed.
- 24** EZYTU17I shows which types of encryption the client will support when receiving data.
- 25** EZYTU17I shows otelnetd sending to the client the type of encryption to use for the data stream (otelnetd to client) and the initial encryption data.

- 26** EZYTU17I shows otelnetd receiving from the client the type of encryption to use for the data stream (client to otelnetd) and the initial encryption data.
- 27** EZYTU17I shows otelnetd acknowledging receipt of the initial encryption data from the client.
- 28** EZYTU17I shows the client acknowledging receipt of the initial encryption data from otelnetd.
- 29** EZYTU17I shows otelnetd verifying its keyids.
- 30** EZYTS08I shows the terminal option negotiation the client has sent/received.
- 31** EZYTU17I shows the client verifying its keyids.
- 32** EZYTU17I shows all data following this command in the data stream (otelnetd to client) will be encrypted using the previously negotiated method of data encryption.
- 33** EZYTU17I shows all data following this command in the data stream (client to otelnetd) will be encrypted via the previously negotiated method of data encryption.
- 34** EZYTU17I traces z/OS UNIX Telnet sending terminal negotiation suboptions to the client.
- 35** EZYTE10I traces the call to tgetent(), which determines client terminal type.
- 36** EZYTO04I shows the user name with which the telnet client logged in.
- 37** EZYTE26E indicates no /etc/banner file was found.
- 38** EZYTS15I and EZYTS16I show that a state change was processed due to options/responses received from the client.
- 39** EZYTY08I traces the parameters passed to the spawned/forked child address space where the OMVS shell runs.
- 40** EZYTE67I traces the socket sets to show whether input/ibits, output/obits, or exception/ebits data has been received.
- 41** EZYTE68I shows exception data received on the parent/child connection.

Cleaning up the utmp entries left from dead processes

Assuming that you have the suggested /etc/rc script, the utmpx file is cleaned up each time the S OMVS command is issued. The utmpx file should not normally need cleaning up, as each terminal slot should be reused the next time someone logs on with that terminal.

Although during normal processing the utmp entries are cleaned up, there are the occasional incidents where zombies are created, or the user may have terminated the session abnormally. When this occurs the utmp entry for that user remains in the /etc/utmpx file until it is cleared out. There is an associated tty reserved for every entry in the /etc/utmpx file including the zombie entries. For dead entries, these ttys will not be available for reuse until someone under superuser erases the /etc/utmpx file.

Note: If you erase the file while someone is logged on, the next logoff will report not finding the utmpx entry for the user. This can be seen with a waitpid failure during that user cleanup.

Chapter 10. Diagnosing Telnet problems

This chapter describes how to diagnosis Telnet problems.

General Telnet server information

The Telnet protocol provides a standardized interface, through which a program on one host (the Telnet client) can access the resources of another host (the Telnet server) as though the client were a local terminal connected to the server host.

Telnet protocol is based on the concept of a Network Virtual Terminal (NVT) and the principle of negotiated options.

An NVT is an imaginary device, providing the necessary basic structures for a standard terminal. Each host client represents an imaginary device with certain terminal characteristics that the host server can support.

The principle of negotiated options is used by the Telnet protocol because many clients and hosts require additional services beyond the base services. Various options can be negotiated. Server and client use a set of conventions to establish operational characteristics for their Telnet connection by means of the DO, DON'T, WILL, WON'T mechanism that is discussed in "Telnet commands and options" on page 380.

Telnet server definitions

Telnet must be defined correctly to both VTAM and TCP/IP. A VTAM APPL definition statement is needed for each Telnet LU that will be used. Model application definitions may also be used. Refer to the *z/OS Communications Server: SNA Resource Definition Reference* for detailed information about these definitions. A corresponding LU must be specified in the BEGINVTAM section of the PROFILE.TCPIP data set. Refer to the *z/OS Communications Server: IP Configuration Reference* for detailed information about these definitions.

Note: All default 3270 LOGMODE entries from the table of Telnet device name parameters in the *z/OS Communications Server: IP Configuration Reference* are for non-SNA sessions. You must code device types and the needed LOGMODE entries for SNA sessions. All default 3270E LOGMODES are for SNA sessions. CLSDST/PASS macros.

Diagnosing Telnet server problems

Problems with Telnet are generally reported under one of the following categories:

- Abends
- Logon problems
- Session hangs
- Incorrect output
- Session outages

Use the information provided in the following sections for problem determination and diagnosis of errors reported against Telnet.

Abends (server)

An abend during Telnet processing should result in messages and error-related information sent to the MVS system console. A dump of the error will be needed unless the symptoms already match a known problem.

Documentation

Code a SYSMDUMP DD or SYSABEND DD statement in the PROC used to start TCP/IP to ensure that a useful dump is obtained in the event of an abend.

Analysis

Refer to *z/OS MVS Diagnosis: Procedures* or see Chapter 3, “Diagnosing abends, loops, and hangs”, for debugging dumps produced during TCP/IP processing.

Logon problems (server)

Telnet login problems are reported when clients are unable to connect to the host application. Generally, this type of problem is caused by an error in the configuration or definitions (either in VTAM or TCP/IP).

If a problem is recreatable, you can use the Telnet DEBUG features. Refer to the *z/OS Communications Server: IP Configuration Guide* for details.

Documentation

The following documentation should be available for initial diagnosis of Telnet login problems:

- Console Log of error messages issued by both TCP/IP and VTAM
- PROFILE.TCPIP data set
- VTAM APPL definitions for Telnet LUs

More documentation that may be needed is discussed in the following analysis section.

Analysis

Table 20 shows symptoms of login problems and refers to the steps needed for initial diagnosis of the error. The information following the chart and associated information can be used for extended diagnosis, if the problem persists.

Table 20. Telnet login problems

Login problem	Analysis steps
No LUs available	1, 2, 6, 10
OPEN failure	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
x-clock (Telnet solicitor panel)	1, 2, 3, 4, 5, 6, 7, 10
x-clock (blank screen)	1, 2, 3, 6, 7, 8, 10, 12
x-clock (application panel)	7, 8, 10
Incorrect USSMSG or DEFAULTAPPL	3, 4, 5, 6, 11

Following are the diagnosis steps referred to in Table 20:

1. Have VTAM APPL definition statements been coded correctly?

Note: There must be a VTAM definition statement or model application name for each LU coded in the PROFILE.TCPIP data set.

2. Is the VTAM node containing the Telnet LU definitions active?
3. Is there a DEFAULTAPPL coded in the PROFILE.TCPIP data set?

4. Is the host application (or DEFAULTAPPL) active?
5. Is there an ALLOWAPPL statement coded that includes the requested application?
6. Have comment delimiters been added or removed as needed in the BEGINVTAM section of the PROFILE.TCPIP data set?
7. Have correct LOGMODEs (or required overrides for SNA) been coded in the PROFILE.TCPIP data set?
8. Does the host application have BIND (session parameter) requirements that are not met by the specified LOGMODE?
9. Is the MSG07 parameter coded in the PROFILE.TCP data set?

Note: MSG07 returns information to the end user indicating the reason for the failure.

10. Are any abends (in VTAM, host application, or TCP/IP) indicated on the MVS system console?

Note: If an abend occurred, refer to the section on abends to continue investigation of the problem.

11. Check the PROFILE.TCPIP data set for the IP to LU mapping.
12. Is an SSL client attempting to connect to a basic port or is a basic client trying to connect to an SSL port?

If the problem still occurs after following the preceding procedure and making any needed changes, obtain the following documentation:

- TCP/IP Services Telnet display output.
- VTAM DISPLAY of Telnet LU.
- VTAM DISPLAY of the target host application.
- Activate DEBUG DETAIL and review additional diagnostic information this function provides.

For information about the Telnet Display command options, refer to the *z/OS Communications Server: IP Configuration Reference*.

The following documentation might also be needed in some cases, but it is suggested that your IBM Software Support Center be contacted before this documentation is obtained:

- TCP/IP packet trace and CTRACE with TELNET option filtered on the IP address of the failing client.
- VTAM buffer trace of the Telnet LU.
- VTAM INTERNAL TRACCE (VIT) with options (API,PIU,MSG,PSS,NRM,SSCP).
- Dump of TCP/IP address space and TCP/IP Dataspace. To capture the necessary areas of storage in the DUMP command, include:
SDATA=(CSA,LSQA,PSA,RGN,SQA,SUM,SWA,TRT,LPA)

To identify the TCP/IP dataspace in the DUMP command include:

DSPNAME=('tcpip_procname'.TCPIPDS1)

For information about obtaining VTAM traces, refer to *z/OS Communications Server: SNA Operation* or to *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT* for your release. Instructions on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids* for your release of MVS.

Session hangs (server)

This section discusses diagnosis of a hang after a session has been successfully connected. A hang would be indicated by the keyboard remaining locked on the client side of the session, with no data being sent to or received from the server host.

If a problem is recreatable, you can use the Telnet DEBUG features. Refer to the *z/OS Communications Server: IP Configuration Guide* for details.

Documentation

To determine the cause of a Telnet session hang, the following documentation is usually required:

- CTRACE specifying the TELNET option filtered on the IP address of the failing client.
- In some cases a VTAM buffer trace of the Telnet LU might be needed.
- Information about what was seen at the client screen.

Analysis

The preceding traces are essential to finding the reason for the session hang. Data entered at the client terminal is sent to the Telnet server on the TCP/IP connection. The TCP/IP packet trace shows the data arriving at or leaving the stack. CTRACE with the option Telnet specified shows the data coming into and out of TELNET (from both the stack and VTAM). Some processing steps during this time are also included in the trace. The CTRACE with TELNET option shows what the TELNET server does with this data.

The VTAM buffer trace shows the data as received by VTAM to be forwarded to the host application. Following the data flow through the traces between VTAM, TCP/IP, and Telnet will provide an indication of where the problem is occurring.

The following list suggests information to check in the traces. Refer to *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT* or to *SNA Network Product Formats* for more information about VTAM buffer trace output.

Was the last activity at the client input or output? If input (data from the client), start with step 1. If output, go to step 9 on page 369.

1. Does the packet trace show data passed to TCP/IP? If not, the problem is in client or emulator code. If data is in the trace, continue with Step 2.
2. Does CTRACE with TELNET option show data passed to Telnet? The TELNET option shows data coming into the TELNET server from the stack and also going out to VTAM (the reverse for outbound data). If not, the error is in the TCP/IP platform code. Otherwise, continue with Step 3.
3. Does VTAM buffer trace show data passed from Telnet? If not, problem is in the Telnet server code. Otherwise, continue with Step 4.
4. Does VTAM buffer trace show data passed to host application? If not, problem is in VTAM code. If buffer trace shows correct data, continue with Step 5.
5. Does the buffer trace show data coming from the host application? If not, the problem is in the host application. Contact your host application support center for these products. Otherwise, continue with Step 6.
6. Does the buffer trace show data sent back to the Telnet LU? If not, the problem is in VTAM. Otherwise, continue with Step 7.
7. Is the last data from the application seen in the CTRACE with TELNET option output? If not, the problem is in the Telnet server. Otherwise, continue with Step 8 on page 369.

8. Does the packet trace show the data sent to the client? If not, the error is in TCP/IP platform. Otherwise, continue with Step 9.
9. Check the data in the packet trace output to see if unlock keyboard is set on in the data stream. If unlock is set in the output data, the problem is in the emulator or client code. Otherwise, continue with Step 10.
10. Check the last data received by the Telnet LU in the VTAM buffer trace. If unlock is set in that data stream, or end bracket or change direction is set in the RH, the problem is in the Telnet server code. If none is set, the host application did not allow for unlocking of the keyboard. You should contact your host application IBM Software Support Center.

If the preceding problem determination shows the error to be in the TCP/IP platform or Telnet server code, a dump will be needed to allow a more detailed investigation of the problem.

Incorrect output (server)

Problems with incorrect output are reported when the data sent to the client is not seen in its expected form. This could be garbled data that is unreadable on the screen, a blank screen when output is expected, or screen formatting problems. These problems are generally traced back to logmode issues. Ensure the primary and alternate screen sizes in the logmode used are correct for the TN3270 or TN3270E emulator that you are using. The logmode coded in the TCPIP profile is suggested to VTAM as the correct logmode for this device type. The VTAM PLU application determines the actual logmode that is used. Therefore this application must be configured correctly to use the appropriate logmode.

If a problem is recreatable, you can use the Telnet DEBUG features. Refer to the *z/OS Communications Server: IP Configuration Guide* for details.

Documentation

Documentation needed to find the source of the error in an incorrect output problem would be:

- CTRACE with TELNET option and the FULLDATATRACE parm active in the profile
- VTAM buffer trace of the Telnet LU
- Client screen output information

Analysis

The main goal of diagnosing this type of problem is to determine if the data was sent incorrectly by the host application or corrupted by VTAM, TCP/IP, Telnet server, or Telnet client code.

Table 21 lists the types of incorrect output that might be seen and the steps needed to identify the code in error.

Table 21. Incorrect output types for Telnet

Incorrect Output	Analysis Steps
Blank screen	1, 6, 7
Garbled or unreadable characters on the screen	2, 3, 4, 5, 6, 7
Incorrectly formatted screen	6, 7

See Table 21 to identify which of the following steps to use in determining the cause of the error.

1. Was the last output data seen in the packet trace displayed at the terminal? If not, the problem is in the client or emulator. Contact your IBM Software Support Center for this product. If the last output was seen at the terminal, go to step 9 on page 369 of the analysis procedure in Session hangs (server), and continue your diagnosis.
2. Was the TELNET command entered with TRANSLATE specified? If so, make sure the translate table is compatible with the capabilities of the client device. If compatible or no TRANSLATE was used, continue with Step 4.
3. Is the data stream sent to the client (packet trace) the same as that sent by Telnet (CTRACE with TELNET option)? The CTRACE with TELNET option entries show the data as it arrived from VTAM and again as it goes to the stack. FULLDATATRACE should be specified when looking for a problem in the data stream.
If not, the problem is with the TCP/IP platform code. Otherwise, continue with Step 4 on page 368.
4. In the data trace output, is the data stream sent by the server the same as received from VTAM? If not, the problem is with the Telnet server code. Otherwise, continue with Step 5 on page 368.

Note: If the client is an ASCII device, these might be different due to EBCDIC-to-ASCII translation. Check the appropriate translate table for compatibility with the client device.

5. In the VTAM Buffer trace with the FULL option specified, is the data in the VTAM USER entry (data received by VTAM) the same as the data in the VTAM BUFF entry (data sent by VTAM)? If not, VTAM has corrupted the code. Otherwise, incorrect data was sent by the application. Contact the IBM Software Support Center for the host application.
6. Is the LOGMODE specified for the negotiated terminal type valid for the actual client device?

Note: A VTAM session display specifying the SID for the session shows the actual logmode selected by the SNA application.

7. Does the device characteristics information in the BIND sent by the host application match the device characteristics information in the specified LOGMODE entry?

Note: This can be checked by comparing the specified LOGMODE entry (refer to *z/OS Communications Server: SNA Customization*) with the BIND in the buffer trace at logon to the selected application. Refer to the *z/OS Communications Server: SNA Programming* for information of the BIND RU as well as SNA Formats.

If the problem is not found after using the analysis steps, contact your IBM Software Support Center for additional diagnostic suggestions.

Session outages (server)

Session outages are reported as an unexpected termination of the TCP/IP connection or the Telnet-to-host application session. A session that has been disconnected or terminated will result in the client being returned to the panel where the initial TELNET command was entered and message EZZ6034I is issued. Refer to *z/OS Communications Server: IP Messages Volume 4 (EZZ-SNM)*.

Telnet sessions can be terminated due to TELNETPARMS specified in the PROFILE.TCPIP data set. The Telnet server ends a session if there is no activity on

the SNA side of the connection for the amount of time specified in the INACTIVE parameter. The Telnet server checks for dormant sessions on the IP side of the connection using the SCANINTERVAL/TIMEMARK parameters specified. When appropriate, the connection is terminated due to this processing. Refer to the SCANINTERVAL/TIMEMARK parameters in the *z/OS Communications Server: IP Configuration Guide* for additional information.

Documentation

The following documentation is needed for initial investigation of problems reported as session outages:

- MVS system console log
- TCP/IP console log
- TCPIP SYSDEBUG data set

Analysis

The preceding output is needed to begin diagnosis of a session outage reported against Telnet. It is also helpful to know what kind of processing the Telnet user was doing at the time of the interrupted session.

The following steps are suggested for initial investigation of a Telnet session outage:

1. If a timeout due to inactivity or termination due to TIMEMARK processing is suspected, check the values set in the PROFILE.TCPIP data set.
2. Additional messages are issued for session outages when the Telnet DEBUG features are active. Refer to the *z/OS Communications Server: IP Configuration Guide* for details of the Telnet DEBUG features.
3. Check the documentation listed in “Documentation” for indications of an error.
 - If the MVS system console indicates a VTAM error, continue diagnosis with your VTAM programmer.
 - If the TCP/IP console shows a Telnet or TCP/IP error, check *z/OS Communications Server: IP Messages Volume 1 (EZA)* and follow the directions for system programmer response for the message.
 - Information in the TCPIP SYSDEBUG data set should contain a detailed description of the error.

If messages are found that do not lead to an accurate diagnosis and resolution of the error, search the APAR data base, available at http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS for more information. If this does not provide a solution, contact the IBM Software Support Center.

4. If only one Telnet user session was affected, continue with step 5. Otherwise, go to step 7 on page 372.
5. If the problem can be recreated by performing the same operation or processing, the following traces should be helpful in further diagnosis of the error:
 - TCP/IP packet trace filtered using the IP address of the failing client, run concurrently with a CTRACE specifying the TELNET option.
 - Component Trace output.
 - VTAM Internal Trace (VIT)
 - VTAM buffer trace output.

Note: Contact your IBM Software Support Center for information about options needed before running these traces.

6. If all Telnet user sessions were interrupted, do one of the following:

- Check the MVS system console and LOGREC for abends. This type of outage is usually seen when an abend in VTAM or the TCP/IP VTAM interface code occurs. See “Abends (server)” on page 366 for information about obtaining and diagnosing a dump of the failure.
 - Check to see if all the TELNET users come in through the same channel interface or through a common router.
7. If there are no messages or abends and all Telnet user sessions have been disconnected, the traces listed in Step 5 on page 371 will be needed during a recurrence of the failure.

A dump of the TCP/IP address space including the TCP/IP dataspace should be taken at this time. To capture the necessary areas of storage in the DUMP command, include:

```
SDATA=(CSA,LSQA,PSA,RGN,SQA,SUM,SWA,TRT,LPA)
```

To identify the TCP/IP dataspace in the DUMP command include:

```
DSPNAME=('tcpip_procname'.TCPDPS1)
```

Instructions on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids* for your release of MVS.

Special considerations when using SSL encryption support

Because data flowing across the connection between the client and the server is encrypted, the data field in the packet trace is also encrypted once SSL handshaking is completed. If problem determination requires seeing Telnet handshake or user data, you also need to run Component Trace to see the decrypted data field. When starting Component Trace, specify **options=(TELNET)** and use IPCS to format the Component Trace. For more information on Component Trace, see Chapter 5, “TCP/IP services traces and IPCS support” on page 53.

The Telnet Component Trace records contain the connection ID in the CID field. The connection ID in the trace corresponds to the connection ID output of the connection display command. Use this field to locate records related to the client in question. Once an LName has been assigned, the Component Trace User field shows the LName, providing additional data for locating your client.

The following Component Trace records might be of interest:

SKSCINIT Succeeded

SSL handshaking completed and subsequent data on this connection is encrypted.

Receive Data from Client

The Data from Client field of this record contains the decrypted data coming from the client.

Send Data to Client

The Data to Client field of this record contains the decrypted data going to the client.

Following is a sample Send Data to Client Component Trace record:

```
MVS181  TELNET  70010004  12:49:06.354966  Send Data to Client
HASID..002A    PASID...002A    SASID..002A    MODID..EZBTTSD
TCB....00000000  REG14...89D37F40  USER...TCPM1011  DUCB...00000000
CID....092552C4  SEQ.....000024BE
...
...
ADDR...00000000  08167AB0  LEN....00000004  Number of Bytes Sent
```

+0000	0000002C					
ADDR...	00000000	7F687950	LEN....	0000002C	Data	to Client	
+0000	F5C1115D	7F1D4011	40401DC8	C9D2D1F5		5A.)".	.HIKJ5
+0010	F6F7F0F0	C140C5D5	E3C5D940	E4E2C5D9		6700A	ENTER USER
+0020	C9C44060	1D4011C1	5013FFEF			ID -.	.A&...

Telnet Component Trace data

To help associate a Component Trace entry with a particular client, the following two Component Trace fields contain data unique to Telnet:

- CID** The connection ID for the connection. This is equivalent to the connection ID output from the connection display command.
- USER** The LUsername associated with the client, once it has been assigned. Prior to LUsername assignment, this field may be null or contain the TCP procedure name. The LUsername is not set until after the completion of the Telnet handshake.

Use these fields in Component Trace formatting to limit the records to be displayed. For example, if you want Telnet records for a client connection ID X'021F' with the LUsername TCPM1011, code the following IPCS command:

```
CTRACE COMP(SYSTCPIP) SUB((TCPIP)) FULL JOBLIST (TCPM1011)
      OPTIONS((TELNET,CID(X'0000021F')))
```

Note: Some of the records pertinent to the connection will not be shown when the output is restricted by the CID and USER options. However, it is often helpful to use the output produced by these filters as a starting point.

General Telnet client information

The Telnet client code runs under TSO in the TSO user's address space. The Telnet client uses the VTAM interface, like other TSO applications, to send data out to the user's terminal.

The Telnet client can run in line mode, when accessing an ASCII host, or run in full-screen mode, if the remote host provides 3270 full-screen support.

Telnet client definitions

The Telnet command must be authorized to be issued by TSO users. Refer to the *z/OS MVS Initialization and Tuning Guide* for information about making Telnet an authorized command. There are no other special definitions or setup requirements to run the Telnet client.

Diagnosing Telnet client problems

Problems that might involve the Telnet client are usually reported as one of the following types:

- Abends
- Session hangs
- Incorrect output

Use the information in the following sections for problem determination and diagnosis of errors reported in the Telnet client.

Abends (client)

An abend in the TELNET client should result in messages and error-related information being sent to the MVS system console. These abends should affect only the TSO user that was running Telnet. A dump of the error is needed unless the symptoms match a known problem.

Documentation

Code a SYSMDUMP DD or SYSABEND DD statement in the TSO PROC to ensure that a useful dump is obtained in the event of an abend. See Chapter 3, “Diagnosing abends, loops, and hangs” on page 21, for more information.

Analysis

Refer to *z/OS MVS Diagnosis: Procedures* or see Chapter 3, “Diagnosing abends, loops, and hangs” on page 21 for more information about debugging dumps produced during TCP/IP processing.

Session hangs (client)

This section discusses diagnosis of a hang after a session has been successfully connected. A hang is indicated by the keyboard remaining locked after sending or receiving data from the remote host.

There are many components involved in the transfer of data from a locally attached device through a Telnet session. Any one of these might be the cause or a contributing factor to the hang. Each must be investigated to define the area responsible for the failure.

Documentation

To determine the cause of a Telnet client session hang, the following is needed:

- Information about what was seen at the client screen
- VTAM buffer trace of the local device LU
- VTAM internal trace (if the error appears to be in VTAM)
- VTAM TSO trace of the user ID issuing Telnet
- GTF trace of SVC93 and SVC94 (TGET/TPUT)
- Telnet client trace
- Dump of the TSO user's address space
- TCP/IP packet trace and CTRACE with TELNET option on remote host (if possible)

The preceding list of documentation is a complete list that includes documentation needed to resolve most types of hangs. All of the indicated data might not be needed for each occurrence of a hang. The following analysis section provides information about what types of data might be needed through each diagnostic step.

Analysis

To assist with diagnosis of a Telnet client hang, it is helpful to be familiar with the components involved and understand which ones interface directly with each other. In the case of a Telnet from an MVS client to a remote host, the following occurs:

- Data is entered by the user and then passed by VTAM to TSO.
- Data is passed from TSO to Telnet client code.
- Data is transferred across the TCP/IP connection to the remote host.
- The remote server sends data to the target application.

Note: It is suggested that a VTAM buffer trace and a Telnet client trace be run while recreating the problem for initial debugging purposes. A sample of the client trace output can be found in Figure 28 on page 377. Refer to *z/OS*

Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures or to *SNA Network Product Formats* for more information about VTAM buffer trace output.

Following are suggested steps for diagnosing a Telnet client hang, along with the documentation needed in each situation.

1. Does the hang affect other Telnet clients? If so, go to “Diagnosing Telnet server problems” on page 365. Otherwise, continue with Step 2.
2. Was the last activity at the terminal input or output? If input, go to step 5. If output, continue with Step 3.
3. Check the data in the VTAM buffer trace to see if unlock keyboard is set on in the data stream. If unlock is set on in the data stream, the problem is in the emulator, control unit, or terminal device. If not, check the Telnet client trace to ensure the output data stream matches what is seen in the buffer trace. If the data streams match, the remote host application has not unlocked the keyboard. Contact your IBM Software Support Center for the host application for more help with the problem. If the data streams do not match, continue with Step 4.
4. The problem appears to be in the VTAM TSO area. Recreate the error while running the Telnet client trace, a GTF trace of SVC93 and SVC94, a VTAM TSO trace, and a VTAM buffer trace. Contact your IBM Software Support Center for assistance in interpreting the traces.
5. Check the VTAM buffer trace to ensure input data was received by VTAM and passed to TSO. If the last data entered at the terminal is not in the VTAM buffer trace, the problem is in the PC emulation code or in the control unit. If input data is correct, continue with Step 6.
6. Is the entered data seen in client trace output? If not, the problem is in VTAM TSO. Follow the instructions in Step 4. If data is in the client trace, the error needs to be diagnosed from the server host. See “Session hangs (server)” on page 368 and follow the path for “last activity at the terminal was input.”

Documentation listed earlier, but not referenced in the previous debugging steps, can be useful in the following situations:

- VTAM internal trace

Note: Data is seen in “BUFF VTAM” VTAM buffer trace entry (entering VTAM from the terminal), but not in the “BUFF USER” VTAM buffer trace entry (passed from VTAM to TSO).

- Dump of TSO user’s address space

Note: Data is seen in the “BUFF USER” VTAM buffer trace entry, but not in the VTAM TSO trace or Telnet client trace.

Contact the IBM Software Support Center for assistance with further diagnosis when data is obtained in these situations.

Note: Information about starting and examining traces is discussed in “Starting Telnet client traces” on page 376.

Incorrect output (client)

Problems with incorrect output are reported when the data seen at the terminal is not in its expected form. This might be garbled data that is unreadable, a blank screen when output is expected, or screen formatting problems.

Documentation

Documentation needed to find the source of the error in an incorrect output problem is:

- VTAM buffer trace of the local device LU
- VTAM TSO trace of the user ID issuing Telnet
- GTF trace of SVC93 and SVC94
- Telnet client trace
- Client screen output information

Analysis

The main goal of diagnosing this type of problem is to determine if the data was sent incorrectly by the host application or was corrupted by the Telnet server, Telnet client, TSO, or VTAM code. The following analysis steps should allow quick determination of whether the problem is a Telnet client problem or must be addressed from the server host.

1. If new data sent to the screen cannot be read (garbled or formatted incorrectly), go to step 4. Otherwise, continue with Step 2.
2. Was the last output data seen in the VTAM buffer trace displayed at the terminal? If not, the problem is in the emulator or device. Contact the appropriate IBM Software Support Center. Otherwise, continue with Step 3.
3. Does the last output data in the Telnet client trace match the data in the VTAM buffer trace? If not, contact your IBM Software Support Center with the client trace, a VTAM TSO trace, and a VTAM buffer trace of the error. Otherwise, this problem must be investigated from the Telnet server side. Continue with the investigation as a Telnet server session hang.
4. Was the TELNET command entered with TRANSLATE specified? If so, make sure the translate table is compatible with the capabilities of the output device. If the table is compatible or no TRANSLATE was used, continue with Step 5.
5. Check the Telnet client trace and VTAM buffer trace. If the data is different, contact your IBM Software Support Center with the client trace, a VTAM TSO trace, and a VTAM buffer trace. Otherwise, continue investigating as a Telnet server incorrect output problem.
6. If the data is formatted incorrectly for the screen size, check the defined session parameters for the negotiated device type for the Telnet server.

If the problem is not found after using the analysis steps, contact your IBM Software Support Center for more diagnostic suggestions.

Telnet client traces

The Telnet client trace shows data received from the remote server to be sent to the local device, and data from the device to be forwarded to the remote host. This includes attention interrupts and some negotiation data seen at the beginning of the session. Data from the initial Telnet negotiation is not seen, only an indication that it is negotiation data and the number of bytes received.

Starting Telnet client traces

Before issuing the Telnet command, the following command should be issued from the TSO “ready” prompt or command line to allocate the trace data set:

```
ALLOC F(DEBUGFIL) DA(data.set.name) NEW
```

Trace data is written to the data set indicated in the command.

The trace is invoked by issuing the Telnet command with the DEBUG option:

```
TELNET hostname (DEBUG
```

Trace example (client)

Figure 28 is sample output from a Telnet client trace showing part of a Telnet login to a remote host.

```
1 EZA8310I DataDelivered; # bytes: 3
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
2 EZA8306I Option neg. stuff arrives
  EZA8310I DataDelivered; # bytes: 6
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8310I DataDelivered; # bytes: 12
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8310I DataDelivered; # bytes: 222
3 EZA8359I Data received from TCP:
4 EZA8361I FF FD 00 FF FB 00 05 C2 11 40 40 1D E4 C5 95 A3 85 99 40 E8
  EZA8361I 96 A4 99 40 E4 A2 85 99 89 84 7A 1D C4 00 00 00 00 00 00
  EZA8361I 00 1D E4 11 C1 50 1D E4 D7 81 A2 A2 A6 96 99 84 7A 1D CC 00
  EZA8361I 00 00 00 00 00 00 00 1D E4 11 C1 F7 1D E4 D5 85 A6 40 97 81
  EZA8361I A2 A2 A6 96 99 84 7A 1D CC 00 00 00 00 00 00 00 00 1D E4 11
  EZA8361I C2 60 1D E4 C1 97 97 93 89 83 81 A3 89 96 95 7A 1D C4 40 40
  EZA8361I 40 40 40 40 40 40 1D E4 11 C3 F0 1D E8 C1 97 97 93 89 83 81
  EZA8361I A3 89 96 95 40 99 85 98 A4 89 99 85 84 4B 40 D5 96 40 C9 95
  EZA8361I A2 A3 81 93 93 81 A3 89 96 95 40 C4 85 86 81 A4 93 A3 40 40
  EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
  EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 00 00 00 11 40 05 13
  EZA8361I FF EF
5 EZA8364I z" w" B" "UEnter Your Userid:D " "" ""U"A&"UPassword:">";
  EZA8364I "U"A7"UNew password:">"U"B-"UApplication:"D
  EZA8364I "U"C0"YApplication required. No Installation Default
  EZA8364I " " "TQ
```

Figure 28. Telnet client trace (Part 1 of 4)

```

EZA8339I In Transparent mode, found IAC at IacOffset 0, CurrentChar is 0
EZA8345I in TelnetRead
EZA8305I in IacNoteArrives
EZA8306I Option neg. stuff arrives
EZA8339I In Transparent mode, found IAC at IacOffset 0, CurrentChar is 3
EZA8345I in TelnetRead
EZA8305I in IacNoteArrives
EZA8306I Option neg. stuff arrives
EZA8339I In Transparent mode, found IAC at IacOffset 214, CurrentChar is 6
EZA8345I in TelnetRead
6 EZA8313I got USERdeliversLINE
EZA8371I in SendData
7 EZA8380I User data is...
EZA8381I 7D '
EZA8381I C2 B
EZA8381I F1 1
EZA8381I 11 "
EZA8381I 40
EZA8381I D4 M
EZA8381I E4 U
EZA8381I E2 S
EZA8381I C5 E
EZA8381I D9 R
EZA8381I F2 2
EZA8381I 11 "
EZA8381I C2 B
EZA8381I 6E >
EZA8381I E3 T
EZA8381I E2 S
EZA8381I D6 0
EZA8381I 40
EZA8381I 40
EZA8381I 40
EZA8381I 40
EZA8381I 40
8 EZA8382I ; Len is 22
EZA8310I DataDelivered; # bytes: 48
EZA8359I Data received from TCP:
EZA8361I 05 C1 11 5D 7F 1D 40 11 40 40 1D C8 C9 D2 D1 F5 F6 F7 F0 F0
EZA8361I C1 40 C5 D5 E3 C5 D9 40 E4 E2 C5 D9 C9 C4 40 60 1D 40 11 C1
EZA8361I 50 13 FF EF 01 C2 FF EF
EZA8364I A)" " " "HIKJ56700A ENTER USERID -" "A&"TQ"B"Q;
EZA8339I In Transparent mode, found IAC at IacOffset 42, CurrentChar is 0
EZA8345I in TelnetRead
EZA8339I In Transparent mode, found IAC at IacOffset 2, CurrentChar is 44
EZA8345I in TelnetRead
EZA8313I got USERdeliversLINE
9 EZA8371I in SendData
EZA8380I User data is...
EZA8381I 7D '
EZA8381I C1 A
EZA8381I D5 N
EZA8381I 11 "
EZA8381I 40
EZA8381I 5A !
EZA8381I A4 u
EZA8381I A2 s
EZA8381I 85 e
EZA8381I 99 r
EZA8381I F3 3
EZA8382I ; Len is 11

```

Figure 28. Telnet client trace (Part 2 of 4)

```

EZA8310I DataDelivered; # bytes: 1106
EZA8359I Data received from TCP:
EZA8361I 05 C3 11 40 40 3C 40 40 40 11 40 40 1D E8 60 60 60 60 60 60
EZA8361I 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
EZA8361I 60 60 60 60 60 40 E3 E2 D6 61 C5 40 D3 D6 C7 D6 D5 40 60 60
EZA8361I 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
EZA8361I 60 60 60 60 60 60 60 60 60 60 60 60 60 60 11 C1 50 1D E8 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 11
EZA8361I C2 60 1D E8 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 11 5B 60 1D E8 D7 C6 F1 61 D7 C6 F1 F3 40 7E 7E
EZA8361I 6E 40 C8 85 93 97 40 40 40 40 D7 C6 F3 61 D7 C6 F1 F5 40 7E
EZA8361I 7E 6E 40 D3 96 87 96 86 86 40 40 40 40 D7 C1 F1 40 7E 7E 6E
EZA8361I 40 C1 A3 A3 85 95 A3 89 96 95 40 40 40 40 D7 C1 F2 40 7E 7E
EZA8361I 6E 40 D9 85 A2 88 96 A6 11 5C F0 1D E8 E8 96 A4 40 94 81 A8
EZA8361I 40 99 85 98 A4 85 A2 A3 40 A2 97 85 83 89 86 89 83 40 88 85
EZA8361I 93 97 40 89 95 86 96 99 94 81 A3 89 96 95 40 82 A8 40 85 95
EZA8361I A3 85 99 89 95 87 40 81 40 7D 6F 7D 40 89 95 40 81 95 A8 40
EZA8361I 85 95 A3 99 A8 40 86 89 85 93 84 11 C3 F3 1D E8 C5 95 A3 85
EZA8361I 99 40 D3 D6 C7 D6 D5 40 97 81 99 81 94 85 A3 85 99 A2 40 82
EZA8361I 85 93 96 A6 7A 11 C4 E3 1D E8 D9 C1 C3 C6 40 D3 D6 C7 D6 D5
EZA8361I 40 97 81 99 81 94 85 A3 85 99 A2 7A 11 C6 D2 1D 60 40 E4 A2
EZA8361I 85 99 89 84 40 40 40 40 7E 7E 7E 6E 11 C6 E2 1D E8 E4 E2 C5
EZA8361I D9 F3 40 40 1D F0 11 C8 F2 1D 60 40 D7 81 A2 A2 A6 96 99 84
EZA8361I 40 40 7E 7E 7E 6E 11 C9 C2 1D 4C 00 00 00 00 00 00 00 00 1D
EZA8361I F0 11 4D F2 1D 60 40 C1 83 83 A3 40 D5 94 82 99 40 7E 7E 7E
EZA8361I 6E 11 4E C2 1D C8 00 00 00 00 00 00 00 00 00 00 00 00 00
EZA8361I 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EZA8361I 00 00 00 00 00 00 1D F0 11 4B D2 1D 60 40 D7 99 96 83 85 84
EZA8361I A4 99 85 40 7E 7E 7E 6E 11 4B E2 1D C8 D4 E5 E2 F4 F2 F2 40
EZA8361I 40 1D F0 11 50 D2 1D 60 40 E2 89 A9 85 40 40 40 40 40 40 7E
EZA8361I 7E 7E 6E 11 50 E2 1D C8 F4 F0 F9 F6 00 00 00 1D F0 11 D2 F2
EZA8361I 1D 60 40 D7 85 99 86 96 99 94 40 40 40 7E 7E 7E 6E 11 D3 C2
EZA8361I 1D C8 00 00 00 1D F0 11 4C C2 1D 60 40 C7 99 96 A4 97 40 C9
EZA8361I 84 85 95 A3 40 40 7E 7E 7E 6E 11 4C D5 1D C8 00 00 00 00 00
EZA8361I 00 00 00 1D F0 11 C9 E2 1D 60 40 D5 85 A6 40 D7 81 A2 A2 A6
EZA8361I 96 99 84 40 7E 7E 7E 6E 11 C9 F5 1D 4C 00 00 00 00 00 00 00
EZA8361I 00 1D F0 11 D7 F3 1D E8 C5 95 A3 85 99 40 81 95 40 7D E2 7D
EZA8361I 40 82 85 86 96 99 85 40 85 81 83 88 40 96 97 A3 89 96 95 40
EZA8361I 84 85 A2 89 99 85 84 40 82 85 93 96 A6 7A 1D 60 11 D9 C7 1D
EZA8361I E8 00 11 D9 C9 1D C8 40 1D F0 60 D5 96 94 81 89 93 1D 60 11
EZA8361I D9 D7 1D E8 00 11 D9 D9 1D C8 40 1D F0 60 D5 96 95 96 A3 89
EZA8361I 83 85 1D 60 11 D9 E8 1D E8 00 11 D9 6A 1D C8 00 1D F0 60 D9
EZA8361I 85 83 96 95 95 85 83 A3 1D 60 11 D9 7A 1D E8 00 11 D9 7C 1D
EZA8361I C8 40 1D F0 60 D6 C9 C4 83 81 99 84 40 1D 60 11 D5 D2 1D 60
EZA8361I 40 C3 96 94 94 81 95 84 40 40 40 7E 7E 7E 6E 11 D5 E2 1D C8
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 1D F0 11 C7 C2 1D 7C 40 E2 85 83 93 81 82 85 93 40 40 40 40
EZA8361I 40 7E 7E 7E 6E 11 C7 D5 1D 7C 40 40 40 40 40 40 40 40 1D F0
EZA8361I 11 C9 C3 13 FF EF

```

Figure 28. Telnet client trace (Part 3 of 4)

```

EZA8364I C - Y ----- TSO/E LOGON --
EZA8364I ----- A& Y
EZA8364I
EZA8364I B- Y
EZA8364I |$- YPF1/PF13==> Help PF3/PF15=
EZA8364I Logoff PA1==> Attention PA2==> Reshow *0 YYou may
EZA8364I request specific help information by entering a '?' in any
EZA8364I entry field C3 YEnter LOGON parameters below: DT YRACF LOGON
EZA8364I parameters: FK - Userid ==> FS YUSER3 0 H2 - Password
EZA8364I ==> IB < 0 (2 - Acct Nbr ==> +B H
EZA8364I 0".K - Procedure==> .S HMVS422 EZA8364I 0
EZA8364I &K - Size ==> &S H4096 0 K2 - Perform ==> LB
EZA8364I H 0 <B - Group Ident ==> <N H 0 IS- New Passw
EZA8364I ord ==> I5 < 0 P3 YEnter an 'S' before each option
EZA8364I desired below: - RG Y RI H 0-Nomail RP Y RR H 0-Nonoti
EZA8364I ce - RY Y R: H 0-Reconnect - R: Y R@ H 0-OIDcard - NK -
EZA8364I Command ==> NS H
EZA8364I 0 GB @ Seclabel
EZA8364I ==>GN @ 0 IC 'Q
EZA8339I In Transparent mode, found IAC at IacOffset 1104, CurrentChar is 0
EZA8345I in TelnetRead
EZA8313I got USERdeliversLINE
EZA8371I in SendData

```

Figure 28. Telnet client trace (Part 4 of 4)

Following are short descriptions of the numbered items in the trace:

- 1** This entry shows the data received from the Telnet server and indicates the number of bytes. The example here is during initial negotiation and does not include the actual data received.
- 2** This indicates the type of data received.
- 3** This entry indicates the data received from TCP (from the Telnet server).
- 4** The actual hexadecimal data received. This trace example is of a transparent mode session, so the data is in EBCDIC. In a line mode session, the data would be in ASCII, and there would be one character per line (like the input data later in the trace).
- 5** This is the translation of the previous hexadecimal data. All hexadecimal characters that translate into readable data are displayed.
- 6** This entry indicates data received from the terminal or PC.
- 7** Following this line is the actual input data. There is a single hexadecimal byte per line that is translated into its readable form.
- 8** This entry follows the input data and indicates the number of bytes received from the terminal.
- 9** This entry indicates the data from the host application (using the Telnet server) that is being sent to the terminal.

Telnet commands and options

For information about Telnet connection negotiations, refer to RFC 2355. Table 22 on page 381 describes the Telnet commands from RFC 854, when the codes and code sequences are preceded by an IAC. For more information about Telnet commands, refer to RFC 854.

Table 22. Telnet commands from RFC 854

Command	Code	Description
SE	X'F0'	End of subnegotiation parameters.
NOP	X'F1'	No operation.
Data Mark	X'F2'	The data stream portion of a Synch. This should always be accompanied by a TCP Urgent notification.
Break	X'F3'	NVT character BRK.
Interrupt Process	X'F4'	The function IP.
Abort output	X'F5'	The function AO.
Are You There	X'F6'	The function AYT.
Erase character	X'F7'	The function EC.
Erase Line	X'F8'	The function EL.
Go ahead	X'F9'	The GA signal.
SB	X'FA'	Indicates that what follows is subnegotiation of the indicated option.
WILL (option code)	X'FB'	Indicates the desire to begin performing, or confirmation that you are now performing, the indicated option.
WON'T (option code)	X'FC'	Indicates the refusal to perform, or continue performing, the indicated option.
DO (option code)	X'FD'	Indicates the request that the other party perform, or confirmation that you are expecting the other party to perform, the indicated option.
DON'T (option code)	X'FE'	Indicates the demand that the other party stop performing, or confirmation that you are no longer expecting the other party to perform, the indicated option.
IAC	X'FF'	Data byte 255.

Table 23 lists the options available for Telnet commands from RFC 1060. For more information about Telnet protocols, refer to RFC 1060 and RFC 1011.

Table 23. Telnet command options from RFC 1060

Option	Option (Hex)	Name
0	0	Binary Transmission
1	1	Echo
2	2	Reconnection
3	3	Suppress Go Ahead
4	4	Approx Message Size Negotiation
5	5	Status
6	6	Timing Mark
7	7	Remote Controlled Trans and Echo
8	8	Output Line Width
9	9	Output Page Size
10	A	Output Carriage-Return Disposition

Table 23. Telnet command options from RFC 1060 (continued)

Option	Option (Hex)	Name
11	B	Output Horizontal Tab Stops
12	C	Output Horizontal Tab Disposition
13	D	Output Formfeed Disposition
14	E	Output Vertical Tabstops
15	F	Output Vertical Tab Disposition
16	10	Output Linefeed Disposition
17	11	Extended ASCII
18	12	Logout
19	13	Byte Macro
20	14	Data Entry Terminal
21	15	SUPDUP
22	16	SUPDUP Output
23	17	Send Location
24	18	Terminal Type
25	19	End of Record
26	1A	TACACS User Identification
27	1B	Output Marking
28	1C	Terminal Location Number
29	1D	Telnet 3270 Regime
30	1E	X.3 PAD
31	1F	Negotiate About Window Size
32	20	Terminal Speed
33	21	Remote Flow Control
34	22	Linemode
35	23	X Display Location
255	FF	Extended-Options-List

Chapter 11. Diagnosing Simple Mail Transfer Protocol (SMTP) problems

The Simple Mail Transfer Protocol (SMTP) is used to transfer electronic mail reliably and efficiently. Recipients of the mail can be users on a local host, users on Network Job Entry (NJE), or users on remote TCP/IP hosts. The SMTPNOTE command is used to send mail to a local or remote host.

This chapter describes how to diagnose problems with SMTP.

Note: For information about diagnosing problems with the other z/OS CS mail application, z/OS UNIX sendmail, see Chapter 12, “Diagnosing z/OS UNIX sendmail and popper problems” on page 393.

Sender SMTP

The sender SMTP performs the following functions:

- Receives notes from the SMTPNOTE CLIST by way of a TSO TRANSMIT command
- Resolves the host name of recipients by way of the RESOLVER module
- Opens a TCP/IP connection with the SMTP server
- Returns mail to the sender, if mail is undeliverable

Receiver SMTP

The receiver SMTP:

- Accepts mail from remote TCP/IP hosts
- Delivers mail to the local user using TSO TRANSMIT to the spool for the local user
- Forwards mail to the next “hop”, if this is not the final destination
- Rejects mail for recipients who are not valid

SMTP environment

Figure 29 shows the SMTP environment.

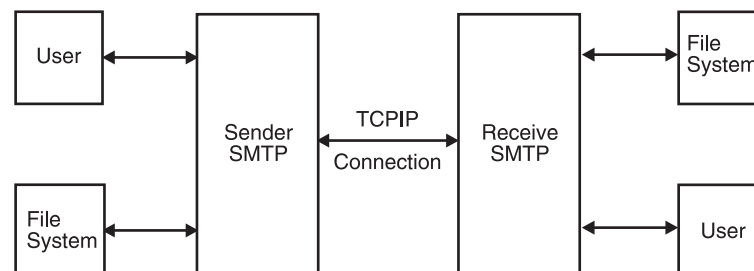


Figure 29. SMTP environment

SMTP definitions

In order to run correctly, SMTP must be defined correctly for both TCP/IP and SMTP. The SMTP.CONFIG and TCPIP.DATA data sets contain the main sender and receiver parameters. The SMTPNOTE CLIST must be customized for your particular installation. The IEFSSNxx member of PARMLIB must be modified to include the following lines:

```
TNF,MVPTSSI  
VMCF,MVPXSSI, nodename (where nodename is the NJE node name)
```

Notes:

1. The NJE node name, *nodename*, must be the same as the *hostname* and the *smtpnode* in the SMTPNOTE CLIST.
2. SMTP can handle only one NJE node name.

Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about configuring SMTP.

Diagnosing SMTP problems

Problems with SMTP are generally reported under one of the following categories:

- Abend
- Spooling
- SMTP does not deliver mail
- SMTP loop
- Mail item has incorrect output

Abends

An abend during SMTP processing should result in messages and error related information being sent to the system console. A dump of the error will be needed unless the symptoms already match a known problem.

Documentation

The following documentation is needed for abends:

- Dump

Note: Code a SYSMDUMP DD or SYSABEND DD statement in the SMTP cataloged procedure to ensure that a useful dump is obtained in the event of an abend.

- Output from the started SMTP procedure
- SYSLOG and LOGREC output for the time of the error

Analysis

Refer to *z/OS MVS Diagnosis: Procedures* or see Chapter 3, “Diagnosing abends, loops, and hangs” on page 21, for information about debugging dumps produced during SMTP processing.

Spooling problems

Spooling problems can occur when the VERB command is being used and the origination information is either missing or not valid. The VERB command requires the originator to have a valid JES user ID and node ID on the SMTP sending system. The originator information is taken from the TSO XMIT (Transmit) command headers.

For more information about the VERB command, refer to *z/OS Communications Server: IP User's Guide and Commands*.

SMTP does not deliver mail

This section discusses diagnosis of mail items that are not delivered to the recipient. Problems with mail not being forwarded can be divided into the following categories:

- Mail not forwarded to a local user
- Mail not forwarded to a user on another NJE host
- Mail not forwarded to remote TCP/IP host

Documentation

The following documentation should be available for initial problem diagnosis:

- TSO console log with the SMTPNOTE messages
- Job log output from the started SMTP procedure
- SMTP.CONFIG data set
- TCPIP.DATA data set

Other documentation that might be needed is discussed in the following section.

Analysis

Use the following procedure to analyze the problem:

1. If the problem is that mail was not forwarded to a local user:
 - a. Was SMTPNOTE customized for your installation?
 - b. Is the local user one that is coded as a restricted user in the SMTP.CONFIG data set?
 - c. Are the JES node parameters coded correctly? This can be determined by issuing a TSO TRANSMIT of a data set to the user and node. If the transmission works, the JES node parameters are coded correctly.
2. If the problem is that a mail note was not forwarded to an NJE host:
 - a. Follow the preceding steps for mail that was not forwarded to a local user.
 - b. Is SMTP configured as an NJE gateway?
 - c. Was SMTPNJE successfully run to create the NJE host table data set?.
 - d. Check whether the NJE node is in the NJE host table data set.

Note: Refer to the *z/OS Communications Server: IP Configuration Reference* for information about SMTP configuration.

3. If the problem is that mail was not forwarded to a remote TCP/IP host:
 - a. Use the SMSMSG SMTP QUEUE command to see the status of the note. Browse the ADDRBLK data set for obvious errors. The ADDRBLK data set is described in "ADDRBLK data set" on page 388.

Note: You should stop SMTP in order to obtain the ADDRBLK data set as it was sent, because the data set is updated during processing and deleted when the number of recipients equals 0.

- b. Has the host name been resolved to an IP address?
Run RESOLVER trace to see if the host name is resolved correctly. The RESOLVER trace is explained in "SMTP RESOLVER trace" on page 390.
- c. Is the remote TCP/IP/SMTP server running?
Use the PING command to see if the remote TCP/IP is running.

Note: Options coded in the SMTP.CONFIG data set directly affect how and when names are resolved by name servers and how often mail delivery is attempted, if there is a problem in the network or the remote NAME server or if the SMTP server is not running.

If the problem still occurs after following this procedure and making any needed changes and corrections, obtain the following documentation and contact the IBM Software Support Center:

- SMTP.CONFIG data set
- TCPIP.DATA data set
- Output from SYSERR and SYSDEBUG of the started SMTP procedure with DEBUG turned on
- ADDRBLK data set

SMTP loop

This section discusses diagnosis of the SMTP address space looping during processing.

Documentation

If SMTP is looping and printing out AMPX... messages to SYSERR, do the following:

- Examine the SYSERR output for AMPX... error messages and traceback information of called routines.
- Call the IBM Software Support Center with this information.

Note: Coding the NOSPIE run-time parameter in the SMTP cataloged procedure might help alleviate a Pascal error recovery loop. For example, code:

```
//SMTP PROC MODULE=SMTP,DEBUG=,PARMS='NOSPIE',SYSERR=SYSERR
```

See Chapter 3, “Diagnosing abends, loops, and hangs” on page 21 for more diagnostic information about diagnosing loops.

Mail item has incorrect output

Problems with incorrect output are reported when the recipient does not see the mail item in its expected form.

Documentation

Use the following documentation to confirm the source of the error:

- SMTP.CONFIG data set
 - TCPIP.DATA data set
 - Output from SYSERR and SYSDEBUG from the started SMTP procedure with DEBUG turned on
 - A packet trace from TCP/IP and network trace facility output
- This documentation might be needed in cases where the actual data in the TCP/IP packets needs to be examined
- ADDRBLK data set.

Note: You should stop SMTP in order to obtain the ADDRBLK data set as it was sent, because the data set is updated during processing and deleted when the number of recipients equals zero.

- SMTPNOTE data set

Analysis

The main goal in diagnosing an incorrect output problem is to determine where the corruption occurs. Is the data corrupted in SMTP, TCP/IP, or by something or someone on the network?

Use the following procedure to analyze the problem:

1. If the problem is that the received mail item has incorrect output:
 - a. Is the correct translation table being used or could it have been customized to cause the error?
Correct the translation error.
 - b. Do TCP/IP and SMTP receive the correct output from the remote host?
Obtain TCP/IP packet trace output or network trace facility output or both to see the actual data in the packets from the remote host.
 - c. Analyze the output from SMTP DEBUG for obvious errors.

Note: The body of the note (mail item) is not shown in this output.

2. If the problem is that the sent mail item has incorrect output:
 - a. Is the correct translation table being used, or could it have been customized to cause the error?
Correct the translation error.
 - b. Was the correct data sent from SMTP or TCP/IP?
Obtain a TCP/IP packet trace to see the actual data in the packets as they leave TCP/IP.
 - c. Analyze the output from SMTP DEBUG for obvious errors.

Note: The body of the note (mail item) is not shown in this output.

If the problem cannot be corrected by this procedure, and you believe that the problem is caused by either SMTP or TCP/IP, call the IBM Software Support Center for further diagnosis.

Forcing resolution of queued mail

Normally, the SMTP server resolves the MX or A records of a piece of mail and stores the mail in the data sets pointed to by the MAILFILEDSPREFIX keyword in the SMTP configuration data set. If the mail cannot be delivered for some period of time, the IP addresses in the mail can become old or obsolete. The data set names for each piece of mail are:

mailfiledsprefix.number.ADDRBLOK
mailfiledsprefix.number.NOTE

There are two ways to force the SMTP server to resolve the addresses. The preferred method is to issue the SMSG SMTP EXPIRE command. Refer to *z/OS Communications Server: IP User's Guide and Commands* for more information about this command.

An alternate method is to modify the ADDRBLK data set for the piece of mail. For each recipient record (records three through the end of the data set), if the first character of the record is an S, then change the S to an E, for expired. This causes SMTP to resolve that record in the ADDRBLK data set the next time the SMTP server is started. To modify the ADDRBLK data set, the data set must be zapped, or a local utility program must be used. The data set cannot be modified using the ISPF editor or IEBUPDATE.

ADDRBLOK data set

An ADDRBLK data set is the master control file for SMTP and is used for tracking the status of a mail item during mail delivery. One ADDRBLK data set is allocated for each piece of mail and is built when the mail is received. The data set is allocated with a high-level qualifier of MAILFILEDSPREFIX from the SMTP.CONFIG data set. The data set is updated during mail processing and is deleted when the number of recipients equals zero.

Note: You might need to stop SMTP in order to obtain the ADDRBLK data set as it was sent, because the data set is updated during processing and deleted when the number of recipients equals zero.

Table 24 shows the format of Record 1 (the master control record) of an SMTP ADDRBLK data set.

Table 24. Format of Record 1 of an SMTP ADDRBLK data set

Characters	Description	Length (in characters)
1–7	Total number of recipients	7
8–14	Number of unresolved recipients	7
15–21	Number of recipients left to send this mail item to	7
22	Unused	1
23–30	File name of note file	8
31	Unused	1
32–39	Date	8
40	Unused	1
41–48	Time	8
49	Unused	1
50–53	Unused	4
54–55	Unused	2
56	Key Value Meaning B BSMTP RPLY file S Spool file M Spool file from Mailer T File from TCP E Error file	1
Note: Characters 57–80 are optional data used only when the key (Character 56) is “S” or “M.”		
57–64	Tag user ID	8
65–72	Tag node ID	8
73–80	Spool ID on the current system	8
77–80	Spool ID of the file source	4

Table 25 on page 389 shows the format of Record 2 (for an unresolved From record) of an SMTP ADDRBLK data set.

Table 25. Format of Record 2 (for an unresolved from record) of an SMTP ADDRBLK data set

Characters	Description	Length (in characters)
1	Key Value Meaning U Unresolved	1
2	Sender path length (user host.domain)	1
3–4	Length of sender ID	2
5–(L1+4)	Sender ID (who sent the mail)	L1
(L1+5) –(L1+6)	Length of sender host.domain	2
(L1+7) –(L1+L2+6)	Sending host.domain	L2
(L1+L2+7)	Length of sender ID	1
(L1+L2+8) –(L1+L2+L3+7)	Sender ID (who sent the mail)	L3

Table 26 shows the format of Record 2 (for a resolved From record) of an SMTP ADDRBLK data set.

Table 26. Format of Record 2 (for a resolved from record) of an SMTP ADDRBLK data set

Characters	Description	Length (in characters)
1	Key Value Meaning M Resolved	1
2	Sender path length (user host.domain)	1
3–4	Length of sender ID	2
5–(L1+4)	Sender ID (who sent the mail)	L1
(L1+5) –(L1+6)	Length of sender host.domain	2
(L1+7) –(L1+L2+6)	Sending host.domain	(L1+L2+7)
(L1+L2+8)	Length of sender ID	1
(L1+L2+9) –(L1+L2+L3+8)	Sender ID (who sent the mail)	L3
(L1+L2+L3+9)	Length of encoded return path	1
(L1+L2+L3+10) –(L1+L2+L3+L4+9)	Encoded return path	L4

Table 27 shows the format of Records 3–*n* of an SMTP ADDRBLK data set.

Table 27. Format of Record 3 (for an unresolved from record) of an SMTP ADDRBLK data set

Characters	Description	Length (in characters)
1	Key Value Meaning U Unresolved M Resolved	1
2–5	Time-to-Live (TTL)	4

Table 27. Format of Record 3 (for an unresolved from record) of an SMTP ADDRBLK data set (continued)

Characters	Description	Length (in characters)
6	Length of return path	1
7–8	Length of recipient user ID	2
9–(L1+8)	Recipient user ID	L1
(L1+9) –(L1+11)	Length of recipient host.domain	2
(L1+12) –(L1+L2+11)	Recipient's host.domain	L2
(L1+L2+12)	Length of recipient path	1
(L1+L2+13) –(L1+L2+L3+12)	Recipient path	L3
(L1+L2+L3+13)	Number of IP addresses	1
(L1+L2+L3+14) –(L1+L2+L3+17)	IP address 1	4
Note: There can be up to 16 IP addresses listed.		

SMTP RESOLVER trace

The RESOLVER trace shows requests and responses sent to and received from name servers. It also shows if local hosts tables are used for name resolution. This trace helps you diagnose problems with host name resolution.

RESOLVER trace output from SMTP is included in the job log output from the started SMTP procedure.

Figure 30 on page 391 shows an example of RESOLVER trace output. Short descriptions of the numbered items in the trace follow the figure.


```

Userid of Caller:      ARMSTRNG
TCP Host Name:        RALVMFE1
Domain Origin:        RALEIGH.IBM.COM
Jobname of TCP/IP:    TCPCS
Communicate Via:      UDP
OpenTimeOut:          30
MaxRetrys:            1
NSPort:               53
NameServer Userid:    NAMESRV
1 NSInternetAddress(.1.) := 9.67.1.5
  NSInternetAddress(.2.) := 9.67.5.44
  Data set prefix used:  TCPCS.BTA1

Resolving Name:       RICKA
Result from InitResolver: OK
Building Name Server Query:
* * * * * Beginning of Message * * * * *
2 Query Id:            1
3 Flags:               0000 0001 0000 0000
  Number of Question RRs: 1
4 Question 1: RICKA.RALEIGH.IBM.COM A IN
  Number of Answer RRs: 0
  Number of Authority RRs: 0
  Number of Additional RRs: 0
  * * * * * End of Message * * * * *
5 Sending Query to Name Server at 9.67.1.5 Result: OK
6 Notification Arrived: UDP data delivered RC = OK
7 UDP Data Length: 55
  Return from WaitForAnswer: OK
  * * * * * Beginning of Message * * * * *
  Query Id:            1
  Flags:               1000 0101 1000 0000
  Number of Question RRs: 1
  Question 1: RICKA.RALEIGH.IBM.COM A IN
  Number of Answer RRs: 1
8 Answer 1: RICKA.RALEIGH.IBM.COM 86400 A IN 9.67.97.3
  Number of Authority RRs: 0
  Number of Additional RRs: 0
  * * * * * End of Message * * * * *
  HostNumber (1) is: 9.67.97.3

```

Figure 30. Example of RESOLVER trace output

Following are short descriptions of numbered items in the trace.

- 1** Address of the name server being used for name resolution. The address is pulled from the TCPIP.DATA data set.
- 2** Identification number of the query. This is also returned in the response and should be used to match queries to responses.
- 3** Bits set to determine the type of query and response. (Refer to RFC 1035.) There are 16 bits (0–15) set in the parameter field of DNS message.

Bit	Meaning
0	Operation: 0=query, 1=response
1–4	Query type: 0=standard, 1=inverse
5	Set if the answer is authoritative
6	Set if the message is truncated
7	Set if recursion is desired
8	Set if recursion is available
9–11	Reserved
12–15	Response type

Value Meaning

- 0 No error
- 1 Format error in query
- 2 Server failure
- 3 Name does not exist
- 4** Actual question sent to the name server
- 5** IP address of the name server being queried
- 6** The response has arrived (UDP in this case)
- 7** Length of the record
- 8** Answer to the question

Chapter 12. Diagnosing z/OS UNIX sendmail and popper problems

This chapter describes how to diagnose problems with z/OS UNIX sendmail, an electronic mail-transport agent and server, and with z/OS UNIX popper, a mail-delivery agent.

Diagnostic aids for sendmail

The following sections describe various tools and techniques available for diagnosing problems with z/OS UNIX sendmail. For a comprehensive discussion of sendmail, refer to the industry-accepted publication *sendmail* by O'Reilly & Associates, Inc. (ISBN 1-56592-222-0). That publication is known throughout the industry as the *bat book*, because of the fruit bat depicted on the cover. This chapter consistently refers to the *bat book* for further information.

You can also find more information about sendmail at the <http://www.sendmail.org> web site.

Note: For information about diagnosing problems with the other z/OS CS mail application, Simple Mail Transfer Protocol (SMTP), see Chapter 11, "Diagnosing Simple Mail Transfer Protocol (SMTP) problems" on page 383.

Debugging switches

The following table is a complete list of debugging switches in sendmail. Some of these switches create long and complex output. Each switch that is especially useful for debugging mail problems is marked "X" in the third column.

Table 28. Debugging switches by category

Category	Bat book reference	Useful for mail problems	Description
-d0.1	37.5.1	X	Print version information
-d0.4	37.5.2	X	Our name and aliases
-d0.10	37.5.3		Operating System defines
-d0.15	37.5.4	X	Dump delivery agents
-d0.20	37.5.5	X	Print network address of each interface
-d0.22	37.5.6		Show uname() failure
-d0.40	37.5.7		Show scanning of interfaces
-d0.44	37.5.8		Print addresses of strings
-d0.90	37.5.9	(obsolete)	Print first 10 rule sets
-d1.1	37.5.10		Show sender information
-d1.5	37.5.11		Dump the sender address
-d2.1	37.5.12		End with finis()
-d2.9	37.5.13		Show file descriptors with <i>dumpfd()</i>
-d3.1	37.5.14		Print the load average
-d3.5	37.5.15		Print load average
-d3.15	37.5.16		Print three load averages

Table 28. Debugging switches by category (continued)

Category	Bat book reference	Useful for mail problems	Description
-d3.20	37.5.17		Show offset for load average
-d3.30	37.5.18		Show result of decision to queue
-d4.80	37.5.19	X	Trace enoughspace()
-d5.4	37.5.20		Tick for queued events
-d5.5	37.5.21		Events set and cleared
-d5.6	37.5.22		Show events triggered
-d6.1	37.5.23	X	Show failed mail
-d6.5	37.5.24		The current error state
-d6.20	37.5.25		Show sender of return to sender
-d7.1	37.5.26		The Queue filename
-d7.2	37.5.27		Show assigned queue filename
-d7.9	37.5.28		Dump file descriptor for the qf file
-d7.20	37.5.29		Show queue names being tried
-d8.1	37.5.30	X	Failure of MX search (low level)
-d8.2	37.5.31	X	Call to getcanonname(3)
-d8.3	37.5.32	X	Trace dropped local hostnames
-d8.5	37.5.33	X	Hostname being tried in getcanonname(3)
-d8.7	37.5.34	X	Yes or no response to -d8.5
-d8.8	37.5.35	X	MX lookup gets wrong type
-d8.20	37.5.36		Inconsistency in returned information
-d9.1	37.5.37		Canonify hostname and RFC1413 queries
-d9.3	37.5.38		Show raw RFC 1413 reply
-d9.10	37.5.39		Show RFC 1413 query being sent
-d10.1	37.5.40		Show recipient delivery
-d10.2	37.5.41		Dump controlling user's address
-d10.5	37.5.42		Show don't send to MeToo address
-d10.100	37.5.43		Predelivery file descriptor dump
-d11.1	37.5.44	X	Trace delivery
-d11.2	37.5.45	X	Show the uid/gid running as during delivery
-d11.20	37.5.46		Show tried D= directories
-d12.1	37.5.47	X	Show mapping of relative host
-d13.1	37.5.48	X	Show delivery
-d13.5	37.5.49		Show addresses that we should not send to
-d13.6	n/a		Trace envelope stripping, dropping, and moving
-d13.10	37.5.50		Trace sendenvelope()
-d13.20	37.5.51		Show final mode
-d13.21	n/a		Show final send queue
-d13.25	n/a		Watch owner deliveries
-d13.29	37.5.52		Show autoqueueing

Table 28. Debugging switches by category (continued)

Category	Bat book reference	Useful for mail problems	Description
-d13.30	37.5.53		Show envelopes being split
-d14.2	37.5.54		Show header field commas
-d15.1	37.5.55		Show network get request activity
-d15.2	37.5.56		Incoming connections
-d15.101	37.5.57		Kernel TCP debugging
-d16.1	37.5.58		Outgoing connections
-d16.101	37.5.59		Kernel TCP debugging
-d17.1	37.5.60		List MX hosts
-d17.9	37.5.61		Show randomizing MX records
-d18.1	37.5.62		Show SMTP replies
-d18.2	37.5.63		Show entry to MAIL From:
-d18.100	37.5.64		Pause on SMTP read error
-d19.1	37.5.65		Show ESMTP MAIL and RCPT parameters
-d20.1	37.5.66	X	Show resolving delivery agent: parseaddr()
-d21.1	37.5.67		Trace rewriting rules
-d21.2	37.5.68	X	Trace \$& macros
-d21.3	37.5.69		Show subroutine calls
-d21.4	37.5.70		Result after rewriting by a rule
-d21.10	37.5.71		Announce failure
-d21.12	37.5.72		Announce success and show LHS
-d21.15	37.5.73		Show \$digit replacement
-d21.35	37.5.74		Show token by token LHS matching
-d21.36	37.5.75		Trace class matching in the LHS
-d22.1	37.5.76	X	Trace tokenizing an address: prescan()
-d22.11	37.5.77	X	Show address before prescan
-d22.12	37.5.78		Show address after prescan
-d22.36	37.5.79		Show each token
-d22.101	37.5.80		Trace low-level state machine
-d24.4	37.5.81		Trace address allocation
-d24.5	37.5.82		Trace assembly of tokens
-d24.6	37.5.83		Show result of buildaddr()
-d25.1	37.5.84	X	Trace "sendtolist"
-d26.1	37.5.85		Trace recipient queueing
-d26.8	37.5.86		Trace self-destructing addresses
-d26.10	37.5.87		Show full send queue in testselfdestruct
-d27.1	37.5.88	X	Trace aliasing
-d27.2	37.5.89	X	Include file, self-reference, error on home
-d27.3	37.5.90	X	Forwarding path and alias wait
-d27.4	37.5.91	X	Print not safe

Table 28. Debugging switches by category (continued)

Category	Bat book reference	Useful for mail problems	Description
-d27.5	37.5.92		Trace aliasing with printaddr()
-d27.8	37.5.93		Show setting up an alias map
-d27.9	37.5.94	X	Show uid/gid changes with :include: reads
-d27.14	37.5.95		Show controlling user that caused change in identity
-d27.20	37.5.96		Show how alias will be looked up in a map
-d28.1	37.5.97	X	Trace user database transactions
-d28.2	37.5.98		Show no match
-d28.4	37.5.99		Show result of lookup
-d28.8	37.5.100		Try hes_getmailhost()
-d28.16	37.5.101		MX records for forward host
-d28.20	37.5.102		Show udp lookup
-d28.80	37.5.103		Preview lookups
-d29.1	37.5.104		Special rewrite of local recipient
-d29.4	37.5.105	X	Trace fuzzy matching
-d29.5	37.5.106		Preview rule set 5
-d29.7	37.5.107		Show overaliasing fuzzy fallback
-d30.1	37.5.108		Trace processing of header
-d30.2	37.5.109		Eat from
-d30.3	37.5.110		Show a to-less header being added
-d30.35	37.5.111		Trace collect states
-d30.94	37.5.112		Trace collect states
-d31.2	37.5.113	X	Trace processing of headers
-d31.6	37.5.114		Is header known?
-d32.1	37.5.115		Show collected headers
-d32.2	37.5.116		Show ARPA mode with setsender
-d33.1	37.5.117		Watch crackaddr()
-d34.1	37.5.118		Watch header assembly for output
-d34.11	37.5.119	X	Trace header generation and skipping
-d35.9	37.5.120	X	Macro values defined
-d35.14	37.5.121		Macro identification
-d35.24	37.5.122		Macro expansion
-d36.5	37.5.123		Trace processing by stab()
-d36.9	37.5.124		Show hash bucket
-d36.90	37.5.125		Trace function applied to all symbols
-d37.1	37.5.126	X	Trace setting of options
-d37.8	37.5.127	X	Trace adding of words to a class
-d38.2	37.5.128	X	Show map opens and failures
-d38.3	37.5.129		Show passes
-d38.4	37.5.130	X	Show result of map open

Table 28. Debugging switches by category (continued)

Category	Bat book reference	Useful for mail problems	Description
-d38.9	37.5.131		Trace map closings and appends
-d38.10	37.5.132		Trace NIS search for end of aliases
-d38.12	37.5.133		Trace map stores
-d38.19	37.5.134	X	Trace switch map finds
-d38.20	37.5.135	X	Trace map lookups
-d38.44	37.5.136		Show nis_getcanonname() record
-d39.1	37.5.137		Display digit database mapping
-d40.1	37.5.138		Trace processing of the queue
-d40.3	37.5.139		Show envelope flags
-d40.4	37.5.140		Show qf file lines as they are read
-d40.8	37.5.141		Show reasons for failure
-d40.9	37.5.142		Show qf and lock file descriptors
-d40.32	37.5.143		Dump the send queue
-d41.1	37.5.144	X	Trace queue ordering
-d41.2	37.5.145		Cannot open qf
-d41.49	37.5.146		Show excluded (skipped) queue files
-d41.50	37.5.147		Show every file in the queue
-d42.2	37.5.148		Show connection checking
-d42.5	37.5.149		Trace caching and uncaching connections
-d43.1	37.5.150		Trace MIME conversions
-d43.3	37.5.151		See the final MIME boundary name
-d43.5	37.5.152		Watch search for boundaries
-d43.8	37.5.153		Show the calculations
-d43.35	37.5.154		Show boundary lines as emitted
-d43.36	37.5.155		Show content transfer encoding
-d43.40	37.5.156		Show parse of Content-Type: header
-d43.99	37.5.157		Print the leading/following comments
-d43.100	37.5.158		Mark collect() and putheader()
-d44.4	37.5.159		Trace safefile()
-d44.5	37.5.160	X	Trace writable()
-d45.1	37.5.161		Show envelope sender
-d45.3	37.5.162		Show saved domain
-d45.5	37.5.163		Show don't send to sender
-d46.9	37.5.164		Show xf file's descriptors
-d48.2	37.5.165	X	Trace calls to the check_rule sets
-d49.1	37.5.166		Trace checkcompat()
-d50.1	37.5.167		Show envelope being dropped
-d50.2	37.5.168		Show Booleans
-d50.10	37.5.169		Also show the send queue

Table 28. Debugging switches by category (continued)

Category	Bat book reference	Useful for mail problems	Description
-d51.4	37.5.170		Show queue entries being unlocked
-d51.104	37.5.171		Prevent unlink of xf file
-d52.1	37.5.172		Show disconnect from controlling TTY
-d52.100	37.5.173		Prevent disconnect from controlling TTY
-d53.99	37.5.174		Trace xclose()
-d54.1	37.5.175		Show error return and output message
-d54.8	37.5.176		Show message and flags
-d55.60	37.5.177		Show file locking
-d56.1	37.5.178		Persistent host status tracing
-d56.2	37.5.179		More persistent host status tracing
-d56.12	37.5.180		Perform a sanity check
-d56.80	37.5.181		Trace creating the path to the status file
-d56.193	37.5.182		Dump MCI record for the host
-d57.2	37.5.183		Monitor vsnprintf() overflows
-d59.1	37.5.184		XLA from contrib
-d60.1	37.5.185	X	Trace map lookups inside rewrite()
-d61.10	37.5.186		Trace gethostbyname()
-d62.1	37.5.187		Log file descriptors before and after all deliveries
-d62.8	37.5.188		Log file descriptors before each delivery
-d62.10	37.5.189		Log file descriptors after each delivery
-d80.1	37.5.190		Content-Length: header (Sun enhancement)
-d81.1	37.5.191		option for remote mode (Sun enhancement)
-d91.100	37.5.192		Log caching and uncaching connections
-d99.100	37.5.193	X	Prevent backgrounding the daemon

Additional diagnostic aids

In addition to debugging switches, you can use the following z/OS UNIX sendmail diagnostic aids:

- Look in the syslog.log for information. Following is a sample z/OS UNIX sendmail syslog.log message:

```
Dec 28 02:13:30 MVS186 sendmail [67108947] : EZZ7514I: sendmail starting
.
Dec 28 02:13:30 MVS186 sendmail [67108947] : starting daemon (8.12.1): SMTP
```

For descriptions of sendmail messages, refer to *z/OS Communications Server: IP and SNA Codes*.

- Use the -v (verbose) command-line switch to print a complete description of all the steps required to deliver a mail message. For details, see *sendmail, 2nd Edition*.

- Use the -X (trace log) command-line switch to record all input, output, SMTP traffic, and other significant transactions into the specified trace file. For details, refer to *sendmail, 2nd Edition*.
- Check the qf file for queueing concerns. z/OS UNIX sendmail stores undeliverable messages in the QueueDirectory that is specified in the configuration file. The QueueDirectory contains data files (df files) named dfxxxxxxx and matching queue-control files (qf files) named qfxxxxxxx. A df file contains the body of a queued message. A qf file holds all the information that is needed to deliver the message. Each queued message has a corresponding df and qf file.

The qf file is line-oriented, containing one item of information per line. The single uppercase character (the code letter) specifies the contents of the line. The complete list of qf code letters is shown in Table 29.

Table 29. qf File code letters

Code	Reference	Meaning	How Many
A	Op Appendix B	AUTH=parameter	Exactly one
B	Bat book 23.9.1	Body type	At most one
C	Bat book 23.9.2	Controlling user	At most one per R line
d	Op Appendix B	directory of df file	Exactly one
D	Bat book 23.9.3	Obsolete	Obsolete
E	Bat book 23.9.4 & Op Appendix B	Obsolete	Obsolete
F	Bat book 23.9.5	Flag bits	Many
H	Bat book 23.9.6	Header definition	Many
I	Bat book 23.9.7	df file's inode number	Exactly one
K	Bat book 23.9.8	Time last processed	Exactly one
M	Bat book 23.9.9	Message (why queued)	At most one
N	23.9.10	Number times tried	At most one
P	Bat book 23.9.11	Priority (current)	At most one
Q	Bat book 23.9.12	Original recipient	At most one
r	Op Appendix B	final recipient	Exactly one
R	Bat book 23.9.13	Recipient address	Many
S	Bat book 23.9.14	Sender address	Exactly one

Table 29. qf File code letters (continued)

Code	Reference	Meaning	How Many
T	Bat book 23.9.15	Time created	Exactly one
V	Bat book 23.9.16	Version	Exactly one
Z	Bat book 23.9.17	DSN envelope ID	At most one
\$	Bat book 23.9.18	Restore macro value	At most one
.	Bat book 23.9.19	End of qf file	At most one

Note: BAT book refers to *sendmail* by O'Reilly & Associates, Inc. (ISBN 1-56592-222-0). Op refers to *Sendmail Installation and Operation Guide* version 8.564 (For Sendmail Version 8.12).

Diagnostic aids for popper

Diagnostic aids for popper are found in the SYSLOGD log information. Following is a sample z/OS UNIX popper log message:

```
Apr 20 14:19:36 MVSU popper[16777240]: Received: "quit"
```

Use the -t trace option to direct all popper message logging to the specified file. The POP server copies the user's entire maildrop to /tmp and then operates on that copy. If the maildrop is particularly large, or inadequate space is available in /tmp, then the server will refuse to continue and terminate the connection.

To test popper, you can mimic a popper client by TELNETing into a popper port (110) and issuing the popper commands documented in RFC 1725. Following are a few of the commands used to verify that popper is listening on port 110:

user *name*

Specifies the mailbox.

pass *string*

Specifies a server/mailbox-specific password.

list [*msg*]

Lists all message numbers and size or information about a specific message.

retr *msg*

Retrieves the specific message to the screen.

quit Closes the connection to popper.

Following is an example of a TELNET exchange:

```
> telnet <host name/ip addr> 110
OK POP (version 2.53) at MVSU.tcp.raleigh.ibm.com starting.

> user user163
OK Password required for USER163

> pass tcpxyz
OK USER163 has 6 messages (4273 octets)
```

```

> list
OK 6 messages (4273 octets)
1 346
2 371
3 333
4 347
5 2541
6 335
.

> retr 3
OK 333 octets
Received: 9BPXR00T@local host by mvsw.tcp.raleigh.ibm.com (8.8.7/8.8.1) id
PAA83
886099 for user163; Tue, 10 Mar 1998 15:36:57 -0500
Date: Tue, 10 Mar 1998 15:36:57 -0500
from USER163 <USER163>USER163
Message-ID: <199803102036.PAA83886099@mvsw.tcp.raleigh.ibm.com>
X-UIDL: 4569e8e12631e857eed8d8b0ca493
Status: 0

hello
.

```

Chapter 13. Diagnosing SNALINK LU0 problems

The TCP/IP host is implemented with the SNALINK LU0 function. This function allows the use of an SNA backbone to transfer TCP/IP protocols. A TCP/IP host with SNALINK LU0 can be an originator, destination, or router for TCP/IP data. To use the SNALINK LU0 function of TCP/IP, each connected host must have VTAM and TCP/IP installed. The SNALINK LU0 application runs in its own address space and is defined as a VTAM application. There are two types of SNALINK implementations:

- SNALINK LU0, which uses VTAM LU0 protocol
- SNALINK LU6.2, which uses VTAM LU6.2 protocol

This chapter describes how to diagnose problems with the SNALINK LU0 function.

Note: SNALINK LU6.2 diagnosis is discussed in Chapter 14, “Diagnosing SNALINK LU6.2 problems” on page 417.

SNALINK LU0 is a very convenient way to connect to TCP/IP hosts using an existing SNA backbone. An IP datagram destined for a remote host that is connected using SNALINK LU0 is passed to the SNALINK LU0 address space by TCP/IP. The data is packaged into an SDLC frame and transmitted to the remote host using SNA LU0 protocol. Two SNALINK LU0 applications can be configured to connect using a single, bidirectional session or with two separate sessions (one dedicated to send data in each direction).

Definitions

The following are required to define a SNALINK LU0:

- Device and link definitions in the TCPIP profile
- Home address and routing information
- VTAM application definitions
- Parameters on the PROC used to start SNALINK LU0

For more information about these required definitions, refer to the *z/OS Communications Server: IP Configuration Reference*.

Problem diagnosis

SNALINK LU0 problems are normally reported as one of the following:

- Abends
- Session hung terminals
- Session outages

Use the information in the following sections for problem determination and diagnosis of errors reported against SNALINK LU0.

When contacting the IBM Software Support Center for any type of SNALINK LU0 problem, have the VTAM application definitions for SNALINK LU0 and the DEVICE and LINK information from the *h/lq.PROFILE.TCPIP* data set for SNALINK LU0.

Abends

An abend for the SNALINK LU0 application should result in messages or error-related information on the MVS system console. Since SNALINK LU0 is a VTAM application, some abends might be generated or first detected by VTAM. These messages indicate that VTAM is abending or a dump is being taken for the SNALINK LU0 application.

Note: In the case of a VTAM error caused by SNALINK LU0, refer to *z/OS Communications Server: SNA Messages* and *z/OS Communications Server: IP and SNA Codes* for initial problem determination.

Documentation

Code a SYSMDUMP DD or SYSABEND DD statement in the SNALINK cataloged procedure.

There are two MVS abends commonly seen during the initialization and startup of the SNALINK LU0 application: X'0C2' and X'0F8'. Both can be caused by the SNALINK LU0 application processing in TCB mode. The VTAM application definition statement for SNALINK LU0 must have the SRBEXIT=YES parameter coded. This should ensure that VTAM passes control to SNALINK LU0 in SRB mode. SNALINK LU0 code has processing that is not allowed in TCB mode. If the SRBEXIT parameter is coded incorrectly or allowed to default, abend X'0C2' or X'0F8' will occur.

Note: Some networking optimizing packages change the defined mode for VTAM applications for performance purposes. It is suggested that this type of program not be used for the SNALINK LU0 application.

Analysis

For more information about debugging abends, refer to Chapter 3, "Diagnosing abends, loops, and hangs" on page 21.

An abend or unexpected termination of the SNALINK LU0 application does not terminate the TCP/IP address space. If there is no alternate route to the remote host, IP datagrams for TCP/IP Services components (such as TELNET and FTP) will not be transmitted until the application is restarted, either manually or using TCP/IP autolog.

Session hangs

This section discusses diagnosis of a hung terminal after a session has been successfully connected. A hang might be detected by TCP/IP users who are connected to the remote system by means of SNALINK LU0 (this could be FTP, TELNET, or other applications).

The SNALINK LU0 application detects a hung terminal if there is no response to data sent. After waiting 30 seconds for a response, SNALINK LU0 ends the session and tries to reestablish the LU-to-LU session with its partner SNALINK LU0 application. This processing is shown on the SNALINK LU0 log or MVS console log.

Documentation

To determine the cause of an SNALINK LU0 session hung terminal, the following might be needed:

- SNALINK LU0 log or MVS console log
- NETSTAT DEVLINKS display output
- VTAM display application status output

- SNALINK LU0 DEBUG trace output
- VTAM buffer trace of the SNALINK LU0 applications
- VTAM internal trace

Note: For information on VTAM traces, refer to *z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures* and *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT*.

This list of documentation includes documentation needed to resolve most types of hung terminals. All of the indicated data might not be needed for each occurrence of a hung terminal. The following section provides information on the types of data that might be needed for each diagnostic step.

Analysis

The first step in analysis is to determine if the SNALINK LU0 is actually hung or if one of the sessions using SNALINK LU0 to transfer data is hung. When the SNALINK LU0 is the only connection between two hosts, an actual hang in the SNALINK LU0 application impacts all data flowing for TCP/IP. This can include TELNET, FTP, and any other application. Use the following steps to help determine the cause of the reported SNALINK LU0 hung terminal:

1. Does all traffic across the SNALINK LU0 stop? A VTAM buffer trace of the SNALINK LU0 application can be used to see if any data is being passed. If data is still flowing on the session, the SNALINK LU0 is not hung. You need to determine which TCP/IP application or component is failing. If there is no data traffic, continue with Step 2.

Note: SNALINK LU0 traffic can also be checked by doing multiple VTAM displays of the SNALINK LU0 application. The SEND and RECEIVE data count should increase for an active session. Often, using the VTAM display to obtain the status of the TRLE might provide useful information.

2. Issue NETSTAT DEVLINKS to determine the status of the SNALINK LU0 TCP/IP device. If the NETSTAT output shows that the application is trying to connect, check the VTAM and SNALINK LU0 consoles for information about a previous error or abend. If NETSTAT indicates "negotiating," verify the session type. You may require a session_type of SINGLE, refer to the *z/OS Communications Server: IP Configuration Reference* for information on configuring session types. If NETSTAT indicates "connected" or "sending," continue with Step 3.
3. At this point, we need to determine the last SNALINK LU0 activity or processing. This is best accomplished with the debug trace. Contact your IBM Software Support Center with information about the last activity from the SNALINK LU0 console and debug trace.

Note: Information on starting and examining the trace data is discussed in "Starting SNALINK LU0 DEBUG trace" on page 413.

Session outages

A session outage is an unexpected abend or termination of the task. Session outages are usually seen only when an irrecoverable error is detected. The error could be a SNALINK LU0 abend or an error return code from a VTAM request. A session outage should not occur without an indication of its cause, either on the SNALINK LU0 or the VTAM console. Since SNALINK LU0 abends have already been discussed separately, this section describes other types of session outages.

For an example of a successful session setup between two SNALINK LU0 applications, refer to the *z/OS Communications Server: IP Configuration Reference*.

Documentation

The following documentation might be needed to determine the source of the error for a session outage problem:

- SNALINK LU0 log
- MVS console log
- VTAM log
- NETSTAT DEVLINKS display output
- VTAM display application status output
- SNALINK LU0 DEBUG trace output
- VTAM buffer trace of the SNALINK
- LU0 applications
- VTAM Internal Trace (VIT)

Note: For information on VTAM traces, refer to *z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures* and *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT*.

Analysis

When a SNALINK LU0 outage occurs, there should be messages and indicators of the reason for the outage. These appear in the SNALINK LU0 log, or on the VTAM console, or both. If an abend has been recorded, continue diagnosis using the section on abends.

The following is an example of a session outage problem. The message EZA5797E Rejecting bind from xxxxx-no DLC found, along with VTAM error message IST663I Bind fail request received, SENSE=080A0000, was displayed on the MVS system console.

Cause: Large packet size sent in a PIU is rejected by the NCP with sense 800A0000 (PIU too long).

Resolution: Reduce the MTU size on this route using the GATEWAY statement.

Traces

The IP packet trace and the SNALINK LU0 DEBUG trace are useful in diagnosing SNALINK LU0 problems.

Using IP packet trace

The IP packet trace facility is used to trace the flow of IP packets. It is useful when tracking the cause of packet loss or corruption. If the LINKNAME parameter of the IP packet trace facility is specified, only packets transferred along the given link are traced. Specifying this parameter is recommended to avoid tracing a large number of unrelated packets. The following command, when passed to the SNALINK LU0 interface, starts the SNALINK LU0 address space packet trace function:

```
MODIFY addr_sp_name,PKTTRACE ON LINKNAME=link_name
```

where addr_sp_name is the name of the local SNALINK LU0 address space and link_name is the VTAM LU name associated with a SNALINK LU0 DEVICE

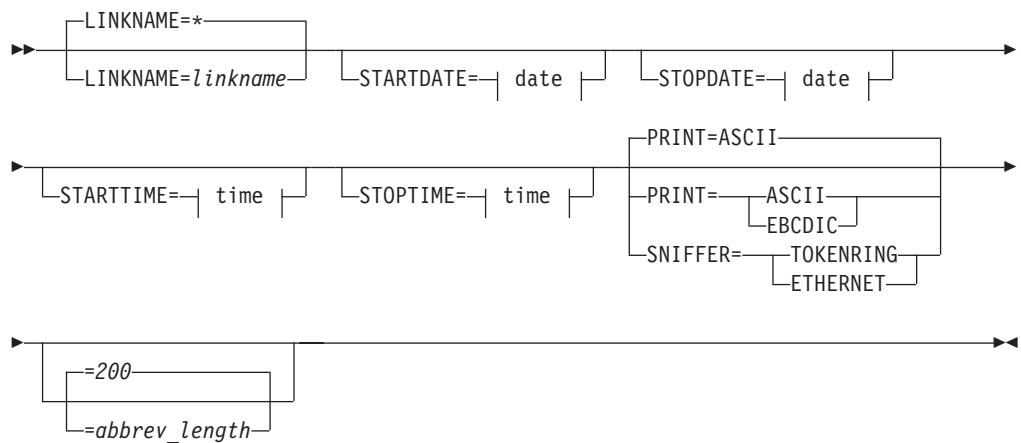
statement defined in the TCPIP profile configuration data set. Refer to the information on the PKTTRACE statement in the *z/OS Communications Server: IP Configuration Reference*.

Formatting a trace report using the TRCFMT utility

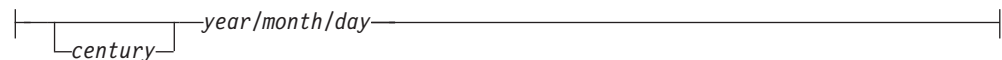
Trace output can be generated by the TRCFMT utility, which uses the GTF data set as input.

The TRCFMT utility has several options that allow the user to select the packets to be formatted and to specify how the trace output is to be formatted. The IP packets that are formatted must meet all the conditions specified by the options passed to the utility. If no options are specified, all packet information stored in the GTF trace data set is formatted.

The syntax of the option for the TRCFMT utility is as follows:



date:



time:



Parameters

LINKNAME

Specifies the name of a link. The actual name specified depends on which device traced the packets. For devices in the TCPIP address space, specify the link name defined on the LINK statement. For SNALINK LU0, specify the LU name defined on the DEVICE statement in the *hlq.PROFILE.TCPIP* data set, rather than the associated link name. For SNALINK LU6.2 and X.25 NPSI, specify the link name on the LINK statement in the appropriate configuration data set. If the LINKNAME option is omitted or an "*" is specified, the options apply to all links.

STARTDATE	<p>Specifies the start date for the trace report. Only packets captured on or after this date appear in the report. The start date refers to the local date of the host where the packets were traced. If this option is omitted, formatting will start with the earliest packet traced.</p> <p>The subfields of the value associated with this option are described below. Each of the subfields has a fixed length and must be supplied with leading zeros.</p> <p>century Specifies the century as a two-digit number</p> <p>year Specifies the year as a two-digit number</p> <p>month Specifies the month as a two-digit number</p> <p>day Specifies the day as a two-digit number</p>
STOPDATE	<p>Specifies the stop date for the trace report. Only packets captured before or on this date appear in the report. The stop date refers to the local date of the host where the packets were traced. If this option is omitted, formatting will stop with the last packet traced.</p> <p>The subfields of the value associated with this option are described above in the description of the STARTDATE option.</p>
STARTTIME	<p>Specifies the start time for the trace report. Only packets captured on or after this time appear in the report. The start time refers to the local time of the host where the packets were traced. If this option is omitted, formatting will start with the earliest time, that is 00:00:00.000000.</p> <p>The subfields of the value associated with this option are described below. Each of the subfields has a fixed length and must be supplied with leading zeros.</p> <p>hour Specifies the hour as a two-digit number</p> <p>min Specifies the minutes as a two-digit number</p> <p>sec Specifies the seconds as a two-digit number</p> <p>microsec Specifies the microseconds as a two-digit number</p>
STOPTIME	<p>Specifies the stop time for the trace report. Only packets captured before or at this time appear in the report. The stop time refers to the local time of the host where the packets were traced. If this parameter is omitted, formatting will stop with the latest time, that is 23:59.59.999999.</p> <p>The subfields of the value associated with this option are described above in the description of the STARTTIME option.</p>
PRINT	<p>Specifies that the trace records are to be formatted as a text report for display on a terminal or printer. This option is the default.</p> <p>ASCII Specifies that the uninterpreted fields in the IP packet are assumed to contain ASCII data. The data in these fields is therefore translated to displayable EBCDIC characters according to the following table before it is presented in the character equivalent dump. The ASCII keyword is the default.</p>

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_
1_
2_	.	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	.	\	.	.	.
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{		}	.	.
8_
9_
A_
B_
C_
D_
E_
F_

EBCDIC

If this keyword is specified, the uninterpreted fields in the IP packet are assumed to contain EBCDIC data. The data in these fields is translated, according to the following table, only to remove nondisplayable EBCDIC characters before it is presented in the character-equivalent dump.

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_
1_
2_
3_
4_	¢	.	<	(+	
5_	&	!	\$	*)	;	¬
6_	-	/	!	,	%	.	>	?
7_	:	#	@	!	=	"
8_	.	a	b	c	d	e	f	g	h	i
9_	.	j	k	l	m	n	o	p	q	r
A_	.	.	s	t	u	v	w	x	y	z
B_
C_	{	A	B	C	D	E	F	G	H	I
D_	}	J	K	L	M	N	O	P	Q	R
E_	\	.	S	T	U	V	W	X	Y	Z
F_	0	1	2	3	4	5	6	7	8	9

SNIFFER

Formats the trace records so that they are suitable for downloading to a DatagLANce Network Analyzer or a Sniffer Network Analyzer for analysis.

TOKENRING

If this keyword is specified, the output is formatted for the token-ring analysis application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on a token ring are displayed. Packets from all devices can be displayed using this option.

ETHERNET

If this keyword is specified, the output is formatted for the ETHERNET analysis application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on an Ethernet will be displayed. Packets from all devices can be displayed using this option.

ETHERNET specifies that a truncated portion of the IP packet is to be formatted. If a length is specified, it must be an integer in the range from 0 to 65535 inclusive and it nominates the maximum number of bytes that are formatted

from each packet. If no length is supplied, the first 200 bytes of each packet are formatted.

As shown in the syntax diagram, the date and time selection criteria can be specified separately. The date and time options are tested independently of each other. Thus, if a date specification is given without a time specification, only the date when the packet was traced is examined to see if it meets the criteria. Similarly, if a time specification is given without a date specification, only the time when the packet was traced is examined to see if it meets criteria. This will, for example, allow the same time slot to be examined over more than one day.

The TRCFMT program uses three optional ddnames to specify the input, output, and error data sets during processing. If data sets other than the default are desired, the ddnames must be assigned to the required data sets when invoking the TRCFMT utility.

The ddnames are:

FMTIN

Identifies the input GTF trace data set. If this ddname is omitted, the default data set is SYS1.TRACE.

FMTOUT

Identifies the destination for the trace report for the PRINT option or the destination of the analyzer records for the SNIFFER option. If this ddname is omitted, the default data set is TCPIP.TRACE.OUTPUT.

The data set associated with this ddname must have the attributes of variable-length records and logical record length of 137 bytes.

FMterr

Identifies the destination for error messages produced when creating a DatagLANce Network Analyzer or a Sniffer Network Analyzer output file in FMTOUT. Reporting of inconsistencies in an IP packet is an integral part of the trace report generated by the PRINT option, but these cannot be sent to FMTOUT when it is to contain data in a format to be read by a DatagLANce Network Analyzer or a Sniffer Network Analyzer. If this ddname is omitted, the default data set is TCPIP.TRACE.ERROR. This ddname is only required for the SNIFFER option.

The data set associated with this ddname must have the attributes of variable-length records and logical record length of 137 bytes.

The TRCFMT utility can be invoked in either the TSO environment or as a batch job using JCL statements.

TRCFMT in the TSO environment: Figure 31 on page 411 shows the steps required to use TRCFMT in the TSO environment.

```

System:  READY
User:    allocate dd(fmtin) ds('sys1.trace')
System:  READY
User:    allocate dd(fmtout) ds(output.trc)
System:  READY
User:    trcfmt linkname=tr1 sniffer=ethernet starttime=08:30 stoptime=17:30
System:  EZA2070W OPENFILE: DDname FMterr not defined by user, using default
System:  EZA2072I OPENDD:   FMterr DDname has been defined to TCPIP.TRACE.ERROR

System:  READY

```

Figure 31. Invoking TRCFMT from TSO

In the figure, TRCFMT reads input from the GTF trace data set SYS1.TRACE, the output is written to the data set userid.OUTPUT.TRC, and error messages are written to the default data set TCPIP.TRACE.ERROR.

```

TRCFMT processing records using the following selection criteria:
Linkname=*                               ABBREV=NONE
Start date=*                             Start time=*
Stop date=*                              Stop time=*
Print = EBCDIC

```

TCPIP Packet Trace formatting routine, Version 3.00 for MVS

```

1
PKT 0000001 DATE=01/02/20 TIME=19:12:43.424633
    TO LINK=SNALU02A          DEV=SNA_LU0
IP   SRC=9.67.21.1          DST=9.67.1.2
    VER=4 HDLEN=5 TOS=X'00' TOTLEN=276 ID=1634 FLAGS=B'000'
    FRAGOFF=0 TTL=64 PROTOCOL=ICMP CHECKSUM=X'4AFF'
ICMP ECHO_REQ              CHECKSUM=X'8B5D' ID=26926 SEQ=58825
DATA LEN=248
    B56D576B 68F9AA09 0671DF5F 1DFFD5F9 0738CE6F *... .9.. ...¼ ..N9 ..¾?
    2278D969 2C097C7F 22CC75D9 1D4CF98F 28941B49 *..R. ..@" .÷.R .<9. .m..
    138B559F 20A239B9 7A7BA0AF 4A724129 1313EABF *.... .s.. :#.. ....
    0098A199 249943CF 4A32CB09 4CAFBBDF 25572D79 *.q.r .r.. .... <...
    7E6A62EF 7A8538E9 405B48FF 29155D59 39A37E0F *=|.. :e.Z $. .. ). .t=.*
    1CA90AC9 1E03121F 5B9AB139 08E9152F 7E6DC0A9 *.z.I .... $. .. .Z.. =.{z*
    5783973F 003EA919 11CFA84F 5832DA89 72A9585F *.cp. .z. .y| ...i .z.¼*
    4BE8C4F9 1EDBB76F 6AE7D869 1B30D57F 231084D9 *.YD9 ...? |XQ. ..N" ..dR*
    1181C28F 5E0C3A49 74C68E9F 17BD68B9 142649AF *.aB. ;... .F.. ....
    5DAF8029 2D0703BF 2786F099 0C1DCCCF 78712A09 *)... .... .f0r ..÷. ....
    4D7EB4DF 39949C79 4BACCB EF 3D80B7E9 4DAA21FF *(=.. .m.. .... .Z (...
    5C9DEC59 0407C70F 1A9DA9C9 64F5CB1F 44EA6039 **..I. ..G. ..zI .5.. ..-.*
    77533E2F 7A177FA9                      *.... :."z

```

```

2
PKT 0000002 DATE=01/02/20 TIME=19:12:43.429125
    FROM LINK=SNALU02A        DEV=SNA_LU0
IP   SRC=9.67.1.2           DST=9.67.21.1
    VER=4 HDLEN=5 TOS=X'00' TOTLEN=276 ID=1593 FLAGS=B'000'
    FRAGOFF=0 TTL=64 PROTOCOL=ICMP CHECKSUM=X'4B28'
ICMP ECHO_REPLY            CHECKSUM=X'935D' ID=26926 SEQ=58825
DATA LEN=248
    B56D576B 68F9AA09 0671DF5F 1DFFD5F9 0738CE6F *... .9.. ...¼ ..N9 ..¾?
    2278D969 2C097C7F 22CC75D9 1D4CF98F 28941B49 *..R. ..@" .÷.R .<9. .m..
    138B559F 20A239B9 7A7BA0AF 4A724129 1313EABF *.... .s.. :#.. ....
    0098A199 249943CF 4A32CB09 4CAFBBDF 25572D79 *.q.r .r.. .... <...
    7E6A62EF 7A8538E9 405B48FF 29155D59 39A37E0F *=|.. :e.Z $. .. ). .t=.*
    1CA90AC9 1E03121F 5B9AB139 08E9152F 7E6DC0A9 *.z.I .... $. .. .Z.. =.{z*
    5783973F 003EA919 11CFA84F 5832DA89 72A9585F *.cp. .z. .y| ...i .z.¼*
    4BE8C4F9 1EDBB76F 6AE7D869 1B30D57F 231084D9 *.YD9 ...? |XQ. ..N" ..dR*
    1181C28F 5E0C3A49 74C68E9F 17BD68B9 142649AF *.aB. ;... .F.. ....
    5DAF8029 2D0703BF 2786F099 0C1DCCCF 78712A09 *)... .... .f0r ..÷. ....
    4D7EB4DF 39949C79 4BACCB EF 3D80B7E9 4DAA21FF *(=.. .m.. .... .Z (...
    5C9DEC59 0407C70F 1A9DA9C9 64F5CB1F 44EA6039 **..I. ..G. ..zI .5.. ..-.*
    77533E2F 7A177FA9                      *.... :."z

```

Figure 32. IP packet trace output for SNALINK LU0

The trace output in Figure 32 was done on the local SNALINK LU0 host. Packets are traced as FROM and TO the local host. This trace output enables the identification of IP packet flow and shows the data content of the packets.

The packets that contain data display the data in hexadecimal digits and, in this case, to their EBCDIC characters.

The message flow for the trace in the Figure is:

- 1** The local host sends an Echo Request packet (packet 1) that is 248 bytes in length, including the ICMP header, to the remote host.
- 2** The remote host returns the local host Echo Request packet (packet 1) as an identical Echo Reply packet (packet 2).

TRCFMT As a batch job: Figure 33 shows a set of JCL statements used to invoke TRCFMT with a batch job.

```
//JOB#1 JOB 'JOB1','TRCFMT BATCH',MSGLEVEL=(1,1),MSGCLASS=X,REGION=4096K,
// CLASS=A
//*-----
//*
//* IKJEFT01 - RUN A TSO COMMAND IN BATCH
//*
//*-----
//IKJEDF01 EXEC PGM=IKJEFT01
//FMTIN DD DSN=SYS1.TRACE,DISP=SHR
//FMTOUT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
TRCFMT STARTTIME=09:00 PRINT=EBCDIC
/*
//
```

Figure 33. Invoking TRCFMT in a batch job

In this example, TRCFMT reads its input from the data set SYS1.TRACE. Its output and error messages are written to system-managed spool files.

Generating a file for the DatagLANce or Sniffer Network Analyzer: The TRCFMT utility supports the generation of a file in a format suitable for downloading to a DatagLANce Network Analyzer or a Sniffer Network Analyzer. This feature is selected with the SNIFFER option, which requires specification of either the ETHERNET or TOKENRING suboption to indicate the format of the file to be generated for the analyzer. Refer to “Formatting a trace report using the TRCFMT utility” on page 407 for information about these suboptions. Once the file is generated, it is downloaded as a binary file to the analyzer and loaded using the standard features of the analyzer. If you are using the Ethernet analyzer application, the DOS file type must be ENC. If you are using the token-ring analyzer application, the DOS file type must be TRC.

SNALINK LU0 DEBUG trace

The SNALINK LU0 DEBUG trace output is written to an internal buffer. The trace can be seen only if a dump of the SNALINK LU0 address space is taken. The trace wraps when the buffer is full (a pointer in the trace header points to the most current entry).

The trace contains information on SNALINK LU0 processing. This includes communication with VTAM and TCP/IP, showing VTAM macro requests and DLC requests.

Starting SNALINK LU0 DEBUG trace

To run the SNALINK LU0 DEBUG trace, SNALINK LU0 must be started with DEBUG listed as the first parameter of the PARM parameter on the EXEC statement of the SNALINK cataloged procedure. For information about this parameter, refer to *z/OS Communications Server: IP Configuration Reference*.

DEBUG trace example

Figure 34 shows part of an internal SNALINK LU0 trace obtained from a dump. As shown in the example, the trace can be located by searching for the characters TRCTBL in the dump of the SNALINK LU0 address space. Following the eyecatcher is the address of the next entry to be written, the starting address of the trace table, and the ending address of the trace table.

Use the information following the trace to interpret the entry types and their meaning.

00012A40	00000000	00000000	E3D9C3E3	C2D34040TRCTBL
00012A50	00018640	00018300	00020000	A020006C	..f ..c.....%
00012A60	00000000	80C15008	94000001	00000000A&.m.....
.					
.					
00018300	B56D355D	F3C92348	00000000	00000000	.._)3I.....
00018310	40000000	00000045	00000000	800091DAj.
00018320	B56D355D	F76940CA	00000000	00000000	.._)7.
00018330	10230000	00012B80	00000000	00000000
00018340	B56D355D	F769568A	00000000	00000000	.._)7.....
00018350	40000000	00000006	00000002	8000E256S.
00018360	B56D355D	F77C748A	00000000	00000000	.._)7@.....
00018370	40000000	00000007	00000001	8000E2A2Ss
00018380	B56D355D	F789628A	00000000	00000000	.._)7i.....
00018390	40000000	0000005A	00000002	8000E2D4!.....SM
000183A0	B56D356C	173D7DC8	E2D5C1D3	E4F0F2C1	.._%. 'HSNALU02A
000183B0	40000000	00000034	00000000	8000CEF88
000183C0	B56D356C	1788D188	E2D5C1D3	E4F0F2C1	.._%.hJhSNALU02A
000183D0	10170000	00024E30	00000000	80000000+.....
000183E0	B56D356C	58C0DACB	00000000	00000000	.._%.{.....
000183F0	17101001	00024E30	00024FC0	080A0000+... {....
00018400	B56D3573	3F9CF1CB	00000000	00000000	.._...1.....
00018410	31000800	091BF1F8	31010207	0000000018.....
00018420	B56D3573	3F9E5C4B	E2D5C1D3	E4F0F2C1	.._....*.SNALU02A
00018430	40000000	00000019	00000000	80009E74
00018440	B56D3573	3FDE2DCB	00000000	00000000	.._.....
00018450	2A100000	00024DC0	00024F80	00000000({..
00018460	B56D3573	3FDEF7CB	E2D5C1D3	E4F0F2C1	.._....7.SNALU02A
00018470	102A0000	00024DC0	00024F80	80000000({..
00018480	B56D3573	3FDF0C8B	E2D5C1D3	E4F0F2C1	.._.....SNALU02A
00018490	40000000	00000034	00000000	8000CEF88
000184A0	B56D3573	401997CB	E2D5C1D3	E4F0F2C1	.._..p.SNALU02A
000184B0	10170000	00024E30	00024FC0	80000000+... {....
000184C0	B56D3573	4019F1CB	E2D5C1D3	E4F0F2C1	.._..1.SNALU02A
000184D0	40000000	00000024	00000000	8000A59Cv.
000184E0	B56D3573	8161A8CB	00000000	00000000	.._a/y.....
000184F0	17100000	00024E30	00024FC0	00000000+... {....
.					
.					
000185F0	23100000	00024F10	00037000	80000118
00018600	B56D3585	4906F5C0	E2D5C1D3	E4F0F2C1	.._e..5{SNALU02A
00018610	02000000	03790011	00037000	84000C60^.....d..-
00018620	B56D3585	49081240	E2D5C1D3	E4F0F2C1	.._e... SNALU02A
00018630	10230000	00024F10	00037000	80009000

00018640.:018FFF.--All bytes contain X'00'

Figure 34. Example of a SNALINK LU0 DEBUG trace

The layout of a SNALINK trace table entry is shown in Table 30.

Table 30. Format of a SNALINK trace table entry

Bytes	Definition
00–07	TOD time stamp
08–0F	LU name, if any
10	Entry Type Value 01 DLC Accept 02 DLC Send 03 DLC Receive 04 DLC Sever 05 DLC Msg Pend Queue Request 06 DLC Msg Pend D-Queue Request 0E MVS DLC emulation 0F DLC Interrupt 10 VTAM Request 17 VTAM OPNDST Exit 1F VTAM CLSDST Exit 22 VTAM SEND Exit 23 VTAM Receive Exit 25 VTAM SESSIONC Exit 2A VTAM OPNSEC Exit 2C VTAM TERMSESS Exit 31 VTAM SCIP Exit 32 VTAM LOSTERM Exit 33 VTAM NSEXIT Exit 34 VTAM TPEND Exit 35 VTAM LOGON Exit 40 SNALINK Internal Message Routine Call
11	DLC Interrupt Code/VTAM RPL REQ Code/ VTAM Receive Exit Chain field
12	VTAM CMD: R15/VTAM Exit: RTNCD
13	VTAM CMD: R0 /VTAM Exit: FDB2/DLC IPRCODE
14–17	RPL Address/DLC MSG ID/TPEND reason code/Internal Message ID
18–1B	VTAM Send/Receive/DLC buffer address/Number of Arguments Passed to Internal Message routine
1C–1F	VTAM Send/Receive/DLC buffer length/Internal Message Routine caller's return address

Chapter 14. Diagnosing SNALINK LU6.2 problems

The SNALINK LU6.2 interface uses the LU type 6.2 protocol to establish a point-to-point connection across a SNA network. SNALINK LU6.2 is capable of establishing a connection with any system that runs TCP/IP and uses the LU type 6.2 protocol.

The SNALINK LU6.2 interface is similar to the SNALINK LU0 and X.25 NPSI interfaces with the connection involving several subsystems. The components of the SNALINK LU6.2 network are shown in Figure 35.

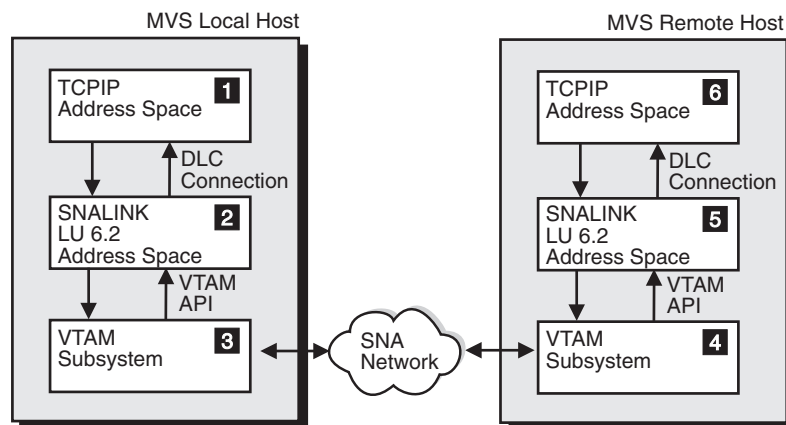


Figure 35. Components of a SNALINK LU6.2 connection on MVS

Following is a brief description of the component interaction and data flow that occurs when data is transferred over a SNALINK LU6.2 network. Each component is cross-referenced to the figure.

- 1** Data is generated and encapsulated on the TCP/IP address space and is passed to the SNALINK LU6.2 address space through a DLC connection.
- 2** The SNALINK LU6.2 address space handles all establishment, aging, and termination of SNA network connections in a manner transparent to the TCP/IP address space. The data is then sent to the local system SNA subsystem. In the case of MVS hosts, this subsystem is VTAM.
- 3** VTAM APPC routines are used to pass the data to the SNA network.
- 4** VTAM routines on the destination system receive the data and pass it through to the SNALINK LU6.2 address space.
- 5** The SNALINK LU6.2 address space sends the data to the TCP/IP address space using a DLC connection.
- 6** The data is unencapsulated and processed by the TCP/IP address space.

Setting up a SNALINK LU6.2 network

Completing the following steps will establish the system described in Figure 35. Each step should be completed in the order given.

This list of steps can be used to diagnose problems in starting components by identifying the prerequisites.

For details about how to complete the steps, refer to the appropriate documentation.

1. Configure the SNALINK LU6.2 network on both the local and remote network hosts. This is fully described in the *z/OS Communications Server: IP Configuration Reference* in the section about configuring and operating the SNALINK LU6.2 interface. The process can be condensed into the following steps:
 - a. Specify SNALINK LU6.2 DEVICE and LINK statements in the *hlq.PROFILE.TCPIP* data set.
 - b. Copy the sample SNALINK LU6.2 cataloged procedure to an authorized data set and update according to your system.
 - c. Define a SNALINK LU6.2 application LU to VTAM.
 - d. Customize a SNALINK LU6.2 configuration data set.
2. Vary the SNALINK LU6.2 VTAM application LUs active on both the local and remote network hosts.
3. Start both the local and remote network TCP/IP address spaces.
4. Start both the local and remote network SNALINK LU6.2 address spaces, if they have not been autologged by the TCP/IP address space.
5. Verify that the network connection has been established between the local host and the remote host. See “Using the SNALINK LU6.2 subcommand” on page 430 for details about how to verify SNALINK LU6.2 connections.

The example in Figure 36 shows the messages that are expected when the SNALINK LU6.2 address space is started and a network connection is established.

```
S SNAL621A
$HASP100 SNAL621A ON STCINRDR
$HASP373 SNAL621A STARTED
1 IEF403I SNAL621A - STARTED - TIME=15.26.03
2 EZA5927I LU62CFG : NO ERRORS DETECTED - INITIALIZATION WILL CONTINUE
3 EZA5932I INITIALIZATION COMPLETE - APPLID: SNAL621A TCP/IP: TCPCS
4 EZA5935I SEND CONVERSATION ALLOCATED FOR 9.67.22.2
5 EZA5933I LINK SNALU62L OPENED
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE SNALU621
4 EZA5936I RECEIVE CONVERSATION ALLOCATED FOR 9.67.22.2
```

Figure 36. Sample MVS System Console Messages on SNALINK LU6.2 Address Space Startup

The following list explains the MVS system console messages on SNALINK LU6.2 address space startup as shown in Figure 36.

- 1** The SNAL621A address space has been started.
- 2** The SNALINK LU6.2 configuration data set for the SNAL621A address space has been successfully parsed.
- 3** The SNAL621A address space displays its local VTAM application LU and the TCP/IP address space name to which it will connect.
- 4** The SNAL621A address space establishes a network connection through the VTAM API.

- 5** The SNALINK LU6.2 address space establishes a DLC connection with its TCP/IP address space.

Common configuration mistakes

Following is a list of common configuration mistakes:

- The SNALINK LU6.2 configuration data set contains a syntax error.
- The SYSTCPD or LU62CFG ddnames in the SNALINK LU6.2 cataloged procedure have been assigned to a data set that is not valid.
- The SNALINK LU6.2 VTAM application LU has not been activated.
- The SNALINK LU6.2 VTAM application LU definition has the option SRBEXIT=YES.
- The SNALINK LU6.2 VTAM application LU definition does not have the option APPC=YES.
- The SNALINK LU6.2 VTAM application LU definition specifies a logon mode table in the MODETAB parameter that does not contain the log mode entry specified in the LOGMODE parameter on the LINK statement in the SNALINK LU6.2 configuration data set. The logon mode entry options used for the local host must be the same as for the remote host.
- The *hlq*.PROFILE.TCPIP data set contains syntax errors in the SNALINK LU6.2 DEVICE, LINK, HOME, GATEWAY, or START statements.
- The maximum buffer size in the SNALINK LU6.2 configuration data set does not match the maximum packet size in the GATEWAY statement of the *hlq*.PROFILE.TCPIP data set.
- The link name in the SNALINK LU6.2 configuration data set does not match the link name on the LINK statement in the *hlq*.PROFILE.TCPIP data set.
- The SNALINK LU6.2 device has not been started by a START statement in the *hlq*.PROFILE.TCPIP data set.
- The user ID assigned to the SNALINK LU6.2 start procedure has not had an OMVS Segment assigned to it using RACF or similar security manager.

Diagnosing problems

SNALINK LU6.2 problems are normally reported under one of the following categories:

- Problems starting the SNALINK LU6.2 address space
- DLC connection
- Network connection establishment
- Network connection loss
- Data loss
- Data corruption

Use the information in the following sections to help you diagnose SNALINK LU6.2 problems.

Quick checklist for common problems

The following list summarizes some initial checks that can be made quickly and are helpful in identifying problem areas:

1. **Is the TCP/IP SNALINK LU6.2 network active?**

PING the remote TCP/IP host from the local TCP/IP host to verify that the SNALINK LU6.2 network is active. If the SNALINK LU6.2 network is not active, continue through this list to identify the problem.

If the PING still fails after working through this list, see “Network connection establishment problems” on page 424 for a detailed list of network connection problems and their solutions.

2. **Have you completed all the required definitions?**

See “Setting up a SNALINK LU6.2 network” on page 417 for the list of definitions and configurations required. Continue through this list if connection problems persist.

3. **Have the VTAM major node and application LU used by the SNALINK LU6.2 address space been varied active?**

See “Useful VTAM operations” on page 431 for details on how to use the VTAM DISPLAY command to identify the status of the VTAM major node and application LU.

If the VTAM application LU is not in a CONCT state, see “Useful VTAM operations” on page 431 for details about how to vary the VTAM application LU active.

4. **Are the TCP/IP and SNALINK LU6.2 devices started and active on the local and remote host?**

Check to see if the TCP/IP and SNALINK LU6.2 devices are active and running. The MVS SDSF facility can be used to view the active address space list for MVS hosts.

If the SNALINK LU6.2 address space will not start, see “Problems starting the SNALINK LU6.2 address space” for a detailed list of startup problems and their solutions.

5. **Did the SNALINK LU6.2 address space list any configuration errors to the SYSPRINT data set?**

Use the JCL DD statement in the SNALINK LU6.2 cataloged procedure to identify the destination of the SYSPRINT output and check for errors. If errors occur, see “Finding error message documentation” on page 439 to determine the reason for the configuration errors. Text in the message documentation will specify the action required to fix the problem.

6. **Have the TCP/IP-to-SNALINK LU6.2 DLC connections been established?**

See “Using NETSTAT” on page 429 for details about how to use the NETSTAT command to identify the status of the DLC connection.

If the status of the DLC connection is not “Connected,” see “DLC connection problems” on page 422 for a detailed list of SNALINK LU6.2 DLC connection problems and their solutions.

7. **Does the MVS system console contain VTAM error messages?**

Refer to *z/OS Communications Server: SNA Messages* and *z/OS Communications Server: IP and SNA Codes* for detailed descriptions of the VTAM error messages and sense codes. These messages might indicate a network configuration or hardware error.

Problems starting the SNALINK LU6.2 address space

Generally, if there is a startup problem, error messages are displayed on the MVS system console during the starting of the SNALINK LU6.2 address space. The address space then terminates.

Documentation

To isolate a SNALINK LU6.2 address space starting problem, note any error messages or abend codes that are displayed on the MVS system console.

Analysis

The following is a list of some of the common SNALINK LU6.2 address space startup problems. Each error symptom is listed with possible causes and resolutions.

- **The message “Errors Detected - Address Space will Terminate” has been displayed on the MVS system console with no other error messages.**

Cause: This error message indicates that an error has occurred with the SNALINK LU6.2 configuration data set.

Resolution: Check the SNALINK LU6.2 SYSPRINT output for messages that tell what kind of syntax error might have occurred. If a syntax error has occurred in the configuration data set, correct it and restart the SNALINK LU6.2 address space.

Refer to the *z/OS Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 configuration data set statement syntax.

- **The message “Error in open of LU62CFG - no data will be read” has been displayed on the MVS system console.**

Cause: The SNALINK LU6.2 address space cannot access a SNALINK LU6.2 configuration data set. The LU62CFG ddname might have been omitted from the SNALINK LU6.2 cataloged procedure.

Resolution: Check the SNALINK LU6.2 cataloged procedure. Ensure that the LU62CFG ddname is assigned a valid SNALINK LU6.2 configuration data set. Refer to the *z/OS Communications Server: IP Configuration Reference* for an example of a SNALINK LU6.2 cataloged procedure.

- **The message “Address Space Already Active - this Address Space will Terminate” has been displayed on the MVS system console.**

Cause: An address space with the same name as the SNALINK LU6.2 address space is already active.

Resolution: Check to see if the address space with the same name is no longer required before stopping it, or rename the SNALINK LU6.2 address space. Restart the SNALINK LU6.2 address space.

- **The messages “Error 0000005A in VTAM OPEN” and “Errors detected in VTAM Initialization - Address Space will terminate” have been displayed on the MVS system console.**

Cause: The SNALINK LU6.2 address space has not been able to find the VTAM application LU that has been defined in the VTAM statement of the SNALINK LU6.2 configuration data set.

Resolution: This problem might be resolved by one or both of the following solutions:

- Check the status of the SNALINK LU6.2 VTAM application LU and its VTAM major node. If it is not in a CONCT state, the VTAM major node and then the VTAM application LU must be activated.

See “Useful VTAM operations” on page 431 for a detailed description of the VTAM operations that display the status of VTAM application LUs and activate them.

- Check the VTAM application LU specified in the VTAM statement of the SNALINK LU6.2 configuration data set. Ensure that it exists and is not duplicated within the domain in which the SNALINK LU6.2 application program resides.

Refer to the *z/OS Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 VTAM statement syntax and the SNALINK LU6.2 VTAM application LU definition.

- **The messages “Error 00000024 in VTAM OPEN” and “Errors detected in VTAM Initialization - Address Space will terminate” have been displayed on the MVS system console.**

Cause: VTAM security is not allowing the SNALINK LU6.2 address space to access the VTAM application LU.

Resolution: Check to see if the SNALINK LU6.2 configuration data set VTAM statement password matches the password set in the VTAM application LU definition and correct it, if necessary.

Refer to the *z/OS Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 VTAM statement syntax and the SNALINK LU6.2 VTAM application LU definition.

- **The SNALINK LU6.2 address space abends with a system abend code of 300 after the “SNALINK LU6.2 address space STARTED” message.**

Cause: The abend code of 300 indicates that there is insufficient storage for the SNALINK LU6.2 address space.

Resolution: Either increase the value of the REGION parameter for the address space or reduce the number of buffers specified in the SNALINK LU6.2 configuration data set. Refer to *z/OS Communications Server: SNA Messages* and *z/OS Communications Server: IP and SNA Codes* for detailed SNALU6.2 abend code descriptions.

- **The SNALINK LU6.2 address space abends with an abend code of S0F8 after the “Initialization Complete...” message.**

Cause: The MVS S0F8 abend code indicates that an SVC was issued in SRB mode. SNALINK LU6.2 is not designed to run with VTAM in SRB mode.

Resolution: The SRBEXIT option in the VTAM application LU definition has been set to “Yes.” Correct the VTAM application LU definition.

Refer to the *z/OS Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 VTAM application LU definition.

If, after following these steps, you do not find the SNALINK LU6.2 startup problem, obtain a description of all abend codes and errors written to the SYSPRINT data set and MVS system console. Most solutions to SNALINK LU6.2 address space starting problems can be solved by reading the error message or abend code descriptions.

See “Finding abend and sense code documentation” on page 439 and “Finding error message documentation” on page 439 for a list of references that contain SNALINK LU6.2 error message and abend code documentation.

DLC connection problems

These problems are related to the TCP/IP DLC connection between the TCP/IP address space and the SNALINK LU6.2 address space.

The DLC connection between the TCP/IP and SNALINK LU6.2 address spaces is established during the SNALINK LU6.2 address space startup after the SNALINK LU6.2 configuration data set has been parsed. This DLC connection can be established independently of the SNA LU type 6.2 connection between two or more SNALINK LU6.2 address spaces. The fundamental requirements of the DLC connection are an active, configured SNALINK LU6.2 address space and an active, configured TCP/IP address space. The DLC connection is initiated by a START statement in *hlq.PROFILE.TCPIP*.

Documentation

To check the status of the DLC connection, perform the following diagnosis steps:

1. Note the SNALINK LU6.2 address space startup messages displayed on the MVS system console.
2. Issue a NETSTAT DEVLINKS command to obtain the status of the DLC connection.

See “Using NETSTAT” on page 429 for details about how to use the NETSTAT command to identify the status of the DLC connection.

If the DLC connection status is not Connected, check the list of common DLC connection problems in the next section.

Analysis

The following list contains some of the common DLC connection problems between the SNALINK LU6.2 address space and the TCP/IP address space. Each error symptom is listed with possible causes and resolutions.

- **The message “Error in DLC connect...” has been displayed on the MVS system console and the NETSTAT DEVLINKS output shows that the DLC connection status is either “Issued Connect” or “Will retry connect.”**

Cause: The TCP/IP address space is attempting to attach to the SNALINK LU6.2 address space, but the SNALINK LU6.2 address space is not responding.

Resolution: Check whether the SNALINK LU6.2 address space is active and start it, if necessary.

- **The SNALINK LU6.2 address space has started, but the “Link open” message has not been displayed on the MVS system console, no other error messages have been displayed on the console, and the NETSTAT DEVLINKS output shows that the DLC connection status is either “Issued Connect” or “Will retry connect.”**

This problem can be due to one of the following situations:

Cause: The SNALINK LU6.2 address space might be rejecting the connect attempt from the TCP/IP address space because it has the wrong TCP/IP ID.

Resolution: Check the SNALINK LU6.2 SYSPRINT output for the “Rejecting DLC path for the link_name, wrong TCP/IP id tcpip_addr_space” error message.

If this error message is displayed, check whether a valid TCPIP.DATA data set was specified as the SYSTCPD ddname in the SNALINK LU6.2 cataloged procedure and correct it, if necessary.

Note: SYSTCPD may be overridden by the global TCPIP.DATA file.

Refer to the *z/OS Communications Server: IP Configuration Reference* for an example of a SNALINK LU6.2 cataloged procedure and for the search order for the TCPIP.DATA data set.

If a valid TCPIP.DATA data set has been used, check the TCP/IP address space specified in the TCPIPJOBNAME statement within it.

Refer to the *z/OS Communications Server: IP Configuration Reference* for a detailed description of the TCPIPJOBNAME statement in the TCPIP.DATA.

Cause: The SNALINK LU6.2 address space might be rejecting the connect attempt from the TCP/IP address space because of a SNALINK LU6.2 link name that is incorrectly defined.

Resolution: Check the SNALINK LU6.2 SYSPRINT output for the “Rejecting DLC path for link_name, not configured” error message.

If this error message is displayed, check to see if the link name specified in the LINK statement of the SNALINK LU6.2 configuration data set matches the link name specified in the LINK statement associated with the SNALINK LU6.2 device defined in *hlq.PROFILE.TCPIP*.

Refer to the *z/OS Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 LINK statement syntax and the TCPIP LINK statement syntax.

- **The SNALINK LU6.2 address space has been started but the “Link opened” message has not been displayed and the NETSTAT DEVLINKS output shows that the DLC connection is “Inactive.”**

Cause: The DLC connection to the SNALINK LU6.2 device associated with the SNALINK LU6.2 address space might not have been started by the TCP/IP address space.

Resolution: Check the START statements in *hlq.PROFILE.TCPIP*.

If the SNALINK LU6.2 device has not been started, use the VARY TCPIP,*procname*,START,*device_name* for the SNALINK LU6.2 device or include the START statement in the *hlq.PROFILE.TCPIP* and restart the TCP/IP address space.

Refer to the *z/OS Communications Server: IP Configuration Reference* for a detailed description of the START statement in the *hlq.PROFILE.TCPIP*.

Network connection establishment problems

These problems are related to the establishment of the SNA LU type 6.2 connection between two or more SNALINK LU6.2 devices.

The SNA LU type 6.2 connection can be established independently of the TCP/IP address space and the DLC link. The fundamental requirements for establishing the LU type 6.2 connection are two active, configured SNALINK LU6.2 devices that have an active SNA network connection between them. There are three ways to initiate the establishment of a network connection.

- Connections with the INIT parameter specified on the DEST statement in the SNALINK LU6.2 configuration data set are established when the SNALINK LU6.2 address space is started.
- Connections with the DATA parameter specified on the DEST statement in the SNALINK LU6.2 configuration data set or connections that have timed out or been terminated are established when a request is made to the SNALINK LU6.2 address space to transfer data across the link.
- Connections can be established using the SNALINK LU6.2 RESTART MODIFY subcommand.

Documentation

To check the status of the LU type 6.2 connection, issue the following MODIFY subcommands to the MVS SNALINK LU6.2 address space.

1. **MODIFY addr_sp_name,LIST,LU=dest_lu_name**

where *addr_sp_name* is the MVS SNALINK LU6.2 address space name and *dest_lu_name* is the SNA destination LU name of the remote SNALINK LU6.2 device.

See “Using the SNALINK LU6.2 subcommand” on page 430 for more information about issuing this command and reading the output.

If the connection status is not “Allocated,” continue with the following commands.

2. **MODIFY addr_sp_name,RESTART,LU=dest_lu_name**

This command attempts to establish the LU type 6.2 connection between the SNALINK LU6.2 devices. During connection establishment, any problems will cause error messages to be output to the MVS system console.

3. **MODIFY addr_sp_name,LIST,LU=dest_lu_name**

If the connection status is still not “Allocated,” note the messages in the SYSPRINT data set and on the MVS system console and continue with the following analysis.

Analysis

The following list contains some of the common SNALINK LU6.2 address space network establishment problems. Each error symptom is listed with possible causes and resolutions.

- **The SNALINK LU6.2 address space issued error message: “Unable to allocate send conversation.”**

Cause: The local VTAM application LU might not be enabled for LU type 6.2 conversations. The name of this LU is specified on the VTAM statement in the SNALINK LU6.2 configuration data set.

Resolution: The APPC option in the VTAM application LU definition must be set to YES to enable LU type 6.2 conversations.

Cause: The remote VTAM application LU names might not identify an LU that is reachable or that can establish an LU type 6.2 conversation over the SNA network. The remote VTAM application LU name is specified in the DEST statement of the SNALINK LU6.2 configuration data set. For dependent LUs, both the SEND and RECV LU names must be able to establish LU type 6.2 conversations.

Resolution: The first step is to check to see if the remote SNALINK LU6.2 device is active. If the remote SNALINK LU6.2 is using VTAM to access the SNA network, see “Useful VTAM operations” on page 431 to check the active status of the remote LU. If the remote SNALINK LU6.2 device is active, use the VTAM error messages to determine why the LU type 6.2 conversation cannot be established with the destination LU. The VTAM error messages are written to the MVS system console immediately before the Unable to allocate send conversation message. VTAM sense code documentation can be found in *z/OS Communications Server: SNA Messages* and *z/OS Communications Server: IP and SNA Codes*. These messages might indicate a network configuration or hardware error.

- **VTAM error message output to the MVS system console: “REQUIRED LOGMODE NAME UNDEFINED.”**

Cause: To allocate LU type 6.2 conversations over an SNA network, both sides must specify matching log modes. The VTAM log modes are defined in log mode tables. The log mode configured for use with this connection cannot be found in the log mode table specified on the VTAM application LU definition.

Resolution: The log mode entry name specified as the LOGMODE parameter on the LINK statement in the SNALINK LU6.2 configuration file must exist in the log mode table specified on the MODETAB statement in the VTAM application LU definition.

Network connection loss problems

SNA network connection loss can be either expected or unexpected. This section deals with unexpected connection problems. The definitions of expected and unexpected losses are discussed before continuing with the analysis for unexpected loss.

Connections for the SNALINK LU6.2 address space can be configured to be normally active or normally inactive. The normally inactive configuration is used when there is a cost involved with the network connection time. Normally inactive connections are expected to experience connection establishment and loss regularly with use. Because of this, the SNALINK LU6.2 address space does not write messages to the MVS system console for connection loss. Connection loss for a normally active connection is unexpected. In this case, the SNALINK LU6.2 address space writes connection loss messages to the MVS system log.

When a connection is configured with the INIT parameter on the DEST statement and a timeout value of zero on the LINK statement in the SNALINK LU6.2 configuration data set, the connection is a normally active connection.

When a connection is configured with the DATA parameter on the DEST statement and a nonzero timeout value on the LINK statement in the SNALINK LU6.2 configuration data set, the connection is a normally inactive connection.

Check the connection experiencing the loss to ensure the loss is unexpected. If the connection loss experienced is specifically caused by errors, the loss is unexpected regardless of the connection configuration.

Documentation

Unexpected connection loss occurs if the SNALINK LU6.2 address space encounters errors that compromise the connection. In this case, error messages are written to the data set specified on the SYSPRINT DD statement in the SNALINK LU6.2 cataloged procedure.

To check the status of the SNA LU type 6.2 connection, issue the LIST MODIFY subcommand to the MVS SNALINK LU6.2 address space. See “Using the SNALINK LU6.2 subcommand” on page 430 for more information about issuing this command and reading the output.

Analysis

Use the error messages in the SNALINK LU6.2 SYSPRINT data set to identify the cause of the loss. See “Finding error message documentation” on page 439 for details on finding the documentation for these messages. Text in the message documentation specifies the action required to fix the problem.

The following is an example of an outage problem.

The message EZA5797E “Rejecting bind from xxxxx-no DLC found”, along with VTAM error message IST663I “Bind fail request received, SENSE=800A0000”, is displayed on the MVS system console.

Cause: Large packet size sent in a PIU is rejected by the NCP with sense 800A0000 (PIU too large).

Resolution: The PIU includes the TH, RH, and RU. SNALINK attempts to send data up to the MAXRU size. The total size of the PIU includes the RU portion and the additional 29 bytes for the TH and RH. If this exceeds the maximum size, NCP

will issue a negative response with sense 800A0000 (PIU too large), which results in the SNA session being taken down between SNALINK and the NCSTLU. When the DLC connection is reestablished, the NCP sends a Bind RU which is then rejected with sense 800A0000. The definitions used in the NCP and SNALINK must be such that MAXRU is at least 29 bytes less than MAXDATA. Refer to *z/OS Communications Server: SNA Network Implementation Guide* for more information on defining the MAXDATA, MAXBFRU, and UNITSZ operands.

Data loss problems

These problems are related to data transfer over the SNALINK LU6.2 network. The first step is to determine the point in the network where the data is being lost. The following information is mainly concerned with determining the actual place of loss.

Documentation

To determine where the data packets are being lost, use the LIST MODIFY command for the SNALINK LU6.2 address space. See “Using the SNALINK LU6.2 subcommand” on page 430 for details. When listing the connection status, the number of packets sent and received over the connection since establishment is displayed in the report. The following steps help you determine the source of the loss.

1. Record the current packet count for the SNALINK LU6.2 devices in the network that support the LIST MODIFY command.
2. Issue the PING command on one end of the connection. In a correctly functioning network, PING sends a data packet to the other end of the connection, which then sends a response data packet back to the PING command.
3. Use the updated packet counts to determine how far the packet went.
4. Issue the PING command from the other end of the connection.
5. Use the updated packet counts to determine how far the packet got.

Note: IP packet trace, as described in “Using IP packet trace” on page 437, can also be used to trace and validate the IP data packets as they enter and leave the SNALINK LU6.2 address space.

Analysis

The following list contains some of the common SNALINK LU6.2 data loss problems. Each error symptom is listed with possible causes and resolutions.

- **Data packets are lost between the TCP/IP and the SNALINK LU6.2 address space (either end).**

This problem can be due to one of the following situations:

Cause: The DLC link between the TCP/IP address space and the SNALINK LU6.2 address space might not be active.

Resolution: See “DLC connection problems” on page 422 to diagnose the DLC link problem.

Cause: The SNALINK LU6.2 address space might be discarding packets.

Resolution: When a condition occurs in the SNALINK LU6.2 address space that causes data to be lost, “discarding datagram” messages are written to the data set specified by the SYSPRINT DD statement in the SNALINK LU6.2 cataloged procedure.

See “Finding error message documentation” on page 439 for details on finding the documentation for these messages. Text in the message documentation specifies the action required to fix the problem.

- **Data packets are actually not lost but the protocol (PING) times out.**

Cause: The SNALINK LU6.2 device might be establishing the LU type 6.2 connection to transfer the data packets. The delay in establishing the connection might be causing the protocol to time out.

Resolution: If the DATA parameter is specified on the DEST statement for the connection in the SNALINK LU6.2 configuration data set, the connection is not established until data is to be transferred over the connection. In this case, after the first data transfer, further data packets will be transferred successfully.

If the TIMEOUT parameter is specified on the LINK statement for the connection in the SNALINK LU6.2 configuration data set, the connection can be timing out too often, causing the connection to be reestablished for each data transfer. In this case, the protocol timeout value or the connection timeout value should be increased.

- **Data packets are lost between the SNALINK LU6.2 devices.**

Cause: The network is failing.

Resolution: Check for VTAM error messages on the MVS system console. See “VTAM buffer traces” on page 439 for more details about using VTAM traces to diagnose the SNA network.

Data corruption problems

To determine the source of corruption for the data packets, use the IP packet tracing facility. This facility traces and validates the IP data packets as they enter and leave the SNALINK LU6.2 address space. Using this facility, the source of corruption can be identified as either the SNA network or the TCP/IP system.

Documentation

Set up the network conditions that are experiencing the data corruption. Start component trace in the SNALINK LU6.2 address space. Use the appropriate amount of data and time to ensure the corruption occurs.

Note: Allocate the MVS GTF trace data set (usually SYS1.TRACE) large enough to hold the expected trace output. This trace data set wraps back to the start of the data set when full, overwriting trace information. When tracing, this option does not collect all the data, which means the corruption could be missed. When formatting, this option turns off some of the IP packet validation processing.

Analysis

The IP packet trace facility analyzes the data corruption problem automatically. Once the trace is collected, the trace data is passed through a formatter, which presents the data packets in an easy-to-read report and validates the contents of the packets against the RFC requirements. Every byte of the data packet is validated including reserved fields. The checksums are also recalculated and verified. If any of the data packets traced are corrupted, the formatter writes messages in the formatted report.

You can use this method, possibly together with TCP/IP internal traces, network level traces, or both, to identify the source and type of corruption.

For details on how to use the IP packet trace facility, see “Using IP packet trace” on page 437.

Documentation references for problem diagnosis

This section contains the information and documentation references required to gather and decode diagnostic information about the SNALINK LU6.2 network connection.

The main tools used for problem diagnosis are the NETSTAT utility, the SNALINK LU6.2 LIST subcommand, VTAM status display operations, the SNALINK LU6.2 internal trace facility, and the IP packet trace facility. The use of these tools is explained in the following sections. An explanation of how to interpret the output from each of these tools is also provided and referenced against the sample output.

For TCP/IP internal tracing or VTAM buffer tracing, you are referred to the appropriate diagnosis documentation.

Two cross-reference sections are provided at the end of this section that list all of the types of abend codes, sense codes, and error messages that can be issued from the SNALINK LU6.2 network connection. For each type of abend code, sense code, or error message, you are referred to the documentation that provides a complete description.

Using NETSTAT

This section describes how to use NETSTAT to query the state of TCP/IP devices. This command can be used to quickly verify the status of the SNALINK LU6.2 device and link with relation to the TCP/IP address space.

The NETSTAT DEVLINKS command output displays only information that is known to TCP/IP.

Note: The TCP/IP address space must be started before the NETSTAT command can query the connection status.

The command NETSTAT DEVLINKS displays the devices and links that have been defined to the main TCP/IP address spaces and the status of these devices (whether or not they are active).

Figure 37 shows a sample of output from the NETSTAT DEVLINKS command.

```
| DevName: SNALU621          DevType: SNALU62
|   DevStatus: Ready
|   LnkName: SNALU62L        LnkType: SNALU62    LnkStatus: Ready
|   NetNum: 1   QueSize: 0
|   BytesIn: 0               BytesOut: 0
|   ActMTU: 32764
|   BSD Routing Parameters:
|     MTU Size: 00000        Metric: 00
|     DestAddr: 0.0.0.0      SubnetMask: 255.0.0.0
|   Multicast Specific:
|     Multicast Capability: No
```

Figure 37. NETSTAT DEVLINKS output example

The example shows four SNALINK LU6.2 devices and associated links known to TCP/IP. The Jobname field contains the name of the SNALINK LU6.2 address space associated with the SNALINK LU6.2 device.

The most significant field for diagnosing DLC connection problems is the Status field. Refer to *z/OS Communications Server: IP Configuration Reference* for detailed interpretation of the device status and its importance in the SNALINK LU6.2 DLC connection.

For more details about the usage, parameters, and output of the NETSTAT command, refer to *z/OS Communications Server: IP System Administrator's Commands*.

Using the SNALINK LU6.2 subcommand

This section details how to use the LIST MODIFY subcommand for the MVS SNALINK LU6.2 address space. The SNALINK LU6.2 address space has interactive commands to control the operation and list the status of the active address space. The LIST MODIFY subcommand writes a report to the MVS system console giving the status of the specified connections.

The connection status listed by the LIST subcommand can be requested for a particular remote VTAM application LU name or destination IP address. The following is an example using the LU parameter:

```
MODIFY procname,LIST LU=lu_name
```

In this example, *procname* is the member name of the cataloged procedure used to start the local SNALINK LU6.2 address space and *lu_name* is the remote VTAM application LU name of the connection for which you are requesting the status.

Figure 38 shows a sample output from the subcommand.

```
f snal621a,list lu=snal622a
EZA5971I LIST ACCEPTED; RANGE = SINGLE CONNECTION
EZA5967I 9.67.22.2 (Connected on 01.051 at 15:44:32)
EZA5968I Connected via: DATA Trace Level: ON
EZA5969I SEND:- Status: Allocated Packets Out: 1
EZA5970I RECV:- Status: Allocated Packets In: 1
EZA5974I LIST COMPLETED
```

Figure 38. LIST MODIFY subcommand output example

An active connection displays the EZA5968I Connected message with the “Allocated” status for both the send and receive conversations.

The SNALINK LU6.2 connection allocates two LU type 6.2 conversations: one for sending data to the remote device and one for receiving data. For independent LUs, the remote LU name is the same for both conversations. For dependent LUs, a remote LU name is specified for both the send and receive conversations.

The **Packets In** and **Packets Out** fields are decimal counters that record the number of data packets received from the remote SNALINK LU6.2 and the number of data packets sent to the remote SNALINK LU6.2, respectively. These fields can be used to identify configuration errors that cause data packets to be lost or discarded. For example, the packet counters can be used to track how far a PING packet travels around the network circuit before it gets lost. Each counter incremented means the packet made it past that point.

For more information about the contents of the messages from the LIST MODIFY subcommand, see the message documentation referenced in “Finding error

message documentation” on page 439. Refer to the *z/OS Communications Server: IP Configuration Reference* for more explanation of the LIST MODIFY subcommand.

Useful VTAM operations

This section describes how to use the VTAM DISPLAY and VARY commands to activate an LU, change an LU definition, and to check the status of an application LU.

VTAM application LUs are defined with VTAM macros in a member of the SYS1.VTAMLST data set. The data set member, called the major node, can contain many application LU definitions, called minor nodes. The application LU names (minor node names) are specified on the VTAM and DEST statements in the SNALINK LU6.2 configuration data set.

Activating an LU

To activate an LU, the major node containing the LU definition must be activated first. If there are no definition errors, all the minor nodes defined in the major node are activated when the major node is activated. If a minor node becomes inactive, it can be activated individually. The following is an example of a VTAM VARY subcommand to activate a major or minor node:

```
VARY NET,ACT,ID=node_name
```

In this example, *node_name* is the major or minor node name to activate.

See “Displaying the status of an LU” for an explanation of the active states for a minor node.

Refer to *z/OS Communications Server: SNA Operation* for a complete description of the VARY ACT subcommand.

Changing an LU definition

To change an LU (minor node) definition, the major node containing the LU definition must be deactivated and then reactivated to force VTAM to read the new definition. The following is an example of a VTAM VARY subcommand to deactivate a major node:

```
VARY NET,INACT,ID=majnode_name
```

In this example, *majnode_name* is the major node name to deactivate.

See “Activating an LU” for the major node activation subcommand.

Refer to *z/OS Communications Server: SNA Operation* for a complete description of the VARY INACT subcommand.

Displaying the status of an LU

To display the status of an LU definition, use the following command:

```
DISPLAY NET,ID=node_name,E
```

In this example, *node_name* is the major or minor node name for which you want to display the status.

Displaying the status of a major node lists all of the minor nodes defined to the major node and their STATUS field. For complete information on status of a minor node, specify the actual minor node name in the command.

The STATUS field for a successfully activated LU definition is set to "CONCT," which means connectable. An LU in this state is waiting for the SNALINK LU6.2 address space to be started. An LU in the CONCT state cannot establish an LU type 6.2 conversation.

Figure 39 shows a sample of the output from an LU in connectable state.

```
D NET,ID=SNAL621A,E
IST097I DISPLAY ACCEPTED
IST075I NAME = NETA.SNAL621A, TYPE = APPL 573
IST486I STATUS= CONCT, DESIRED STATE= CONCT
IST1447I REGISTRATION TYPE = CDSERV
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=***NA*** USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=LU62MODE USS LANGTAB=***NA***
IST1632I VPACING = 0
IST597I CAPABILITY-PLU INHIBITED,SLU INHIBITED,SESSION LIMIT NONE
IST231I APPL MAJOR NODE = SNALNK1A
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST1500I STATE TRACE = OFF
IST271I JOBNAME = ***NA***, STEPNAME = ***NA***, DSPNAME = ***NA***
IST228I ENCRYPTION = OPTIONAL , TYPE = DES
IST1563I CKEYNAME = SNAL621A CKEY = PRIMARY CERTIFY = NO
IST1552I MAC = NONE MACTYPE = NONE
IST1050I MAXIMUM COMPRESSION LEVEL - INPUT = 0, OUTPUT = 0
IST1633I ASRCVLM = 1000000
IST1634I DATA SPACE USAGE: CURRENT = ***NA*** MAXIMUM = ***NA***
IST171I ACTIVE SESSIONS = 0000000000, SESSION REQUESTS = 0000000000
IST172I NO SESSIONS EXIST
IST314I END
```

Figure 39. DISPLAY subcommand output example for connectable LU

After the SNALINK LU6.2 address space has successfully started, the STATUS field is set to ACTIV, which means in use by an address space.

Figure 40 on page 433 shows a sample of the output from a DISPLAY command for an LU in active state.

```

D NET,ID=SNAL621A,E
IST097I DISPLAY ACCEPTED
IST075I NAME = NETA.SNAL621A, TYPE = APPL 545
IST486I STATUS= ACT/S, DESIRED STATE= ACTIV
IST1447I REGISTRATION TYPE = CDSERVR
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=***NA*** USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=LU62MODE USS LANGTAB=***NA***
IST1632I VPACING = 0
IST597I CAPABILITY-PLU ENABLED ,SLU ENABLED ,SESSION LIMIT NONE
IST231I APPL MAJOR NODE = SNALNK1A
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST1500I STATE TRACE = OFF
IST271I JOBNAME = SNAL621A, STEPNAME = SNAL621A, DSPNAME = IST84E76
IST228I ENCRYPTION = OPTIONAL , TYPE = DES
IST1563I CKEYNAME = SNAL621A CKEY = PRIMARY CERTIFY = NO
IST1552I MAC = NONE MACTYPE = NONE
IST1050I MAXIMUM COMPRESSION LEVEL - INPUT = 0, OUTPUT = 0
IST1633I ASRCVLM = 1000000
IST1634I DATA SPACE USAGE: CURRENT = 0 MAXIMUM = 136
IST171I ACTIVE SESSIONS = 0000000003, SESSION REQUESTS = 0000000000
IST206I SESSIONS:
IST634I NAME STATUS SID SEND RECV VR TP NETID
IST635I SNAL622A ACTIV-S EAABEEC3D9B90C37 0002 0000 0 0 NETA
IST635I SNAL622A ACTIV/SV-S EAABEEC3D9B90C35 0002 0002 0 0 NETA
IST635I SNAL622A ACTIV-P F6ABEEC3DCB90B7D 0000 0002 0 0 NETA
IST314I END

```

Figure 40. DISPLAY subcommand output example for active LU

This example shows that the SNALINK LU6.2 address space (SNAL621A) has been started successfully and has its local LU (SNAL621A) in use with three sessions active to a remote LU (SNAL622A).

For each SNALINK LU6.2 connection, VTAM establishes three sessions between the application LUs. The first is the control session, which is the middle session in the example. The other two sessions are established for the LU type 6.2 conversations allocated for the connection, one for sending data and one for receiving data.

Refer to *z/OS Communications Server: SNA Operation* for more information about the DISPLAY command.

Traces

The following traces can be used to obtain information about the data flows and actions of the SNALINK LU6.2 network connection. The SNALINK LU6.2 internal trace is the most useful for determining the state of the SNALINK LU6.2 address space. The IP packet trace facility is the most helpful trace facility for monitoring IP packets transferred across the SNALINK LU6.2 network. The TCP/IP internal traces can be used to diagnose problems with the DLC link between TCP/IP and SNALINK LU6.2. The VTAM buffer trace is used to monitor data transactions through the VTAM API interface.

Using SNALINK LU6.2 internal traces

The SNALINK LU6.2 internal traces are written to the location specified by the SYSPRINT statement in the SNALINK LU6.2 cataloged procedure. These traces provide information on the internals of the SNALINK LU6.2 address space.

SNALINK LU6.2 internal tracing is enabled by specifying the following statement in the SNALINK LU6.2 configuration data set:

```
TRACE DETAIL ALL
```

The SNALINK LU6.2 internal trace can also be started by passing a MODIFY console command to the SNALINK LU6.2 interface. The following MODIFY command will start internal tracing:

```
MODIFY procname,TRACE DETAIL ALL
```

In this example, *procname* is the member name of the cataloged procedure used to start the local SNALINK LU6.2 address space.

Refer to the *z/OS Communications Server: IP Configuration Reference* for detailed descriptions of the TRACE statement parameters and the TRACE subcommand parameters.

```

EZA5925I TCP/IP ADDRESS SPACE NAME SET TO TCPCS
EZA5926I LU62CFG : STARTING PASS 1 OF 2
EZA5926I LU62CFG : STARTING PASS 2 OF 2
1 EZA5927I LU62CFG : NO ERRORS DETECTED - INITIALIZATION WILL CONTINUE
EZA5932I INITIALIZATION COMPLETE - APPLID: SNAL621A TCP/IP: TCPCS
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
7 EZA5994I CNOS FOR INDEPENDENT PARTNER; SESSLIM=0002, WINNER=0001, LOSER=0001
EZA5984I OLU= SNAL621A, DLU= SNAL622A, IP ADDRESS= 9.67.22.2
EZA6029E OPRCNOS ERR. R15 00000000 R0 0000000B RTNCD 00000000 FDBK2 0000000B
EZA6030E OPRCNOS ERR. RCPRI= 0008, RCSEC= 0001
EZA6031E OPRCNOS SENSE CODE RECEIVED: 08570003
EZA6032E SENSE CODE SPECIFIED: 00000000
EZA6023E UNABLE TO COMPLETE CNOS ON LU SNAL622A FOR 9.67.22.2
2 EZA6009W CONVERSATIONS FOR 9.67.22.2 TERMINATED
EZA6011E UNABLE TO ALLOCATE SEND CONVERSATION FOR 9.67.22.2
1 EZA5933I LINK SNALU62L OPENED
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
EZA5936I RECEIVE CONVERSATION ALLOCATED FOR 9.67.22.2
EZA5986I VTAM CONVERSATION ALLOCATED; CONVID= 01000011, SID= F6ABEEC3DCB90B89
EZA5984I OLU= SNAL622A, DLU= SNAL621A, IP ADDRESS= 9.67.22.2
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
2 EZA5935I SEND CONVERSATION ALLOCATED FOR 9.67.22.2
EZA5986I VTAM CONVERSATION ALLOCATED; CONVID= 01000012, SID= EAABEEC3D9B90C3F
EZA5984I OLU= SNAL621A, DLU= SNAL622A, IP ADDRESS= 9.67.22.2
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
4 EZA5992I IP DATAGRAM ADDED TO THE VTAM SEND QUEUE, LENGTH= 276,
QUEUE COUNT = 1
EZA5985I LU= SNAL622A, LINKNAME= SNALU62L, IP ADDRESS = 9.67.22.2
3 EZA5988I VTAM SENT LOGICAL RECORD; CONVID= 01000012,
SID= EAABEEC3D9B90C3F, LENGTH= 280
EZA5984I OLU= SNAL621A, DLU= SNAL622A, IP ADDRESS= 9.67.22.2
EZA5995I NUMBER OF IP PACKETS SENT ON 9.67.22.2 = 1
6 EZA5999I 450001140315000040014D4C09431601094301020800D58BEFBF74
0AB56D2E96EF8F85CA03EFDD80
EZA5999I 78B6FE8035FE858058EE968017DAAD8015B4AE80092C55803DA14680271B7D807DCC5
E8074502580
EZA5999I 39CDF68000F24D8023BE0E8012A9F5805434A6804C9F1D806249BE805779C5807B955
6800961ED80
EZA5999I 7C2F6E800DF958006B006800E7ABD801C2F1E80597B65803444B6800B298D805508
CE80352D3580
EZA5999I 2B13668006AE5D80217C7E807455058079DC168060492D80644A2E804232D580175E
C6804F39FD80
EZA5999I 6831DE802206A580625B768062C0CD805FF38E806F10758021922680021D9D80664F3
E805C904580
EZA5999I 03C2D6806C906D807E04EE8075C615801FAD868039593D8011D49E801DF1E5807412
368000000000
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
3 EZA5989I VTAM RECEIVED LOGICAL RECORD; CONVID= 01000011,
SID= F6ABEEC3DCB90B89, LENGTH= 280
EZA5984I OLU= SNAL622A, DLU= SNAL621A, IP ADDRESS= 9.67.22.2
EZA5996I NUMBER OF IP PACKETS RECEIVED ON 9.67.22.2 = 1
6 EZA5999I 4500011402E8000040014D7909430102094316010000DD8BEFBF74
0AB56D2E96EF8F85CA03EFDD80
EZA5999I 78B6FE8035FE858058EE968017DAAD8015B4AE80092C55803DA14680271B7D807DCC5
E8074502580
EZA5999I 39CDF68000F24D8023BE0E8012A9F5805434A6804C9F1D806249BE805779C5807B955
6800961ED80
EZA5999I 7C2F6E800DF958006B006800E7ABD801C2F1E80597B65803444B6800B298D805508C
E80352D3580
EZA5999I 2B13668006AE5D80217C7E807455058079DC168060492D80644A2E804232D580175E
C6804F39FD80
EZA5999I 6831DE802206A580625B768062C0CD805FF38E806F10758021922680021D9D80664F3
E805C904580

```

Figure 41. SNALINK LU6.2 internal trace output (Part 1 of 2)

```

EZA5999I 03C2D6806C906D807E04EE8075C615801FAD868039593D8011D49E801DF1E5807412
368000000000
5 EZA5990I IP DATAGRAM PACKED INTO MESSAGE, LENGTH= 276
EZA5985I LU= SNAL622A, LINKNAME= SNALU62L, IP ADDRESS = 9.67.22.2
EZA5938I RECEIVED OPERATOR SHUTDOWN REQUEST
2 EZA5937I LINK SNALU62L CLOSED
3 EZA5989I VTAM RECEIVED LOGICAL RECORD; CONVID= 01000011,
SID= F6ABEEC3DCB90B89, LENGTH= 280
EZA5984I OLU= SNAL622A, DLU= SNAL621A, IP ADDRESS= 9.67.22.2
EZA5996I NUMBER OF IP PACKETS RECEIVED ON 9.67.22.2 = 1
6 EZA5999I 4500011402E8000040014D7909430102094316010000DD8BEFBF74
0AB56D2E96EF8F85CA03EFDD80
EZA5999I 78B6FE8035FE858058EE968017DAAD8015B4AE80092C55803DA14680271B7D807DCC5
E8074502580
EZA5999I 39CDF68000F24D8023BE0E8012A9F5805434A6804C9F1D806249BE805779C5807B955
6800961ED80
EZA5999I 7C2F6E800DFF958006B006800E7ABD801C2F1E80597B65803444B6800B298D805508C
E80352D3580
EZA5999I 2B13668006AE5D80217C7E807455058079DC168060492D80644A2E804232D580175E
C6804F39FD80
EZA5999I 6831DE802206A580625B768062C0CD805FF38E806F10758021922680021D9D80664F3
E805C904580
EZA5999I 03C2D6806C906D807E04EE8075C615801FAD868039593D8011D49E801DF1E5807412
368000000000
5 EZA5990I IP DATAGRAM PACKED INTO MESSAGE, LENGTH= 276
EZA5985I LU= SNAL622A, LINKNAME= SNALU62L, IP ADDRESS = 9.67.22.2
EZA5938I RECEIVED OPERATOR SHUTDOWN REQUEST
2 EZA5937I LINK SNALU62L CLOSED

```

Figure 41. SNALINK LU6.2 internal trace output (Part 2 of 2)

Following are brief explanations of the numbered items in the output:

- 1 Messages written to the MVS system console.
- 2 VTAM send and receive conversation status.
- 3 Information about the VTAM API interface data flow.

The VTAM interface information contains the LU type 6.2 conversation ID (Convid), the VTAM session ID (SID), length of the VTAM logical record, the origin and destination VTAM application LUs, and the home IP address.

The VTAM logical record length should be four greater than the length of the TCP/IP datagram packet to account for the VTAM logical record header.
- 4 Information about data received from the TCP/IP DLC connection.

Datagrams received from the SNALINK LU6.2 DLC connection are unpacked from the DLC message and added to the appropriate VTAM send queue for transmission.
- 5 Information about data received from the VTAM API interface.

Datagrams received from the VTAM API are packed into a DLC message buffer.
- 6 Hexadecimal display of data passed through the SNALINK LU6.2 address space.

There should be a hexadecimal display for every 4 and 5 event.
- 7 Change number of sessions (CNOS) data.

Refer to the *z/OS Communications Server: SNA Programmer's LU 6.2 Guide* for more information about CNOS processing.

Using IP packet trace

The IP packet trace facility is used to trace the flow of IP packets. It is useful when tracking the cause of packet loss or corruption.

If the LINKNAME parameter of the IP packet trace facility is specified, only packets transferred along the given link are traced. Specifying this parameter is recommended to avoid tracing a large number of unrelated packets. The following command, when passed to the SNALINK LU6.2 interface, starts the SNALINK LU6.2 address space packet trace function:

```
MODIFY procname,PKTTRACE ON LINKNAME=link_name
```

In this example, *procname* is the member name of the cataloged procedure used to start the local SNALINK LU6.2 address space and link_name is the local TCP/IP host SNALINK LU6.2 link name.

See Chapter 5, “TCP/IP services traces and IPCS support” on page 53 for details about how to use the IP packet trace facility.

Figure 42 on page 438 shows an example of a CTRACE formatted packet trace record.

```

19 VIC127  PACKET  00000001 22:52:18.648744 Packet Trace
To Link      : SNALU62L      Device: SNA_LU6.2      Full=276
Tod Clock    : 2001/02/20 22:52:18.648743
Lost Records : 0              Flags: Pkt Ver2 Out
Source Port  : 0              Dest Port: 0      Asid: 01F6 TCB: 007AEE88
IpHeader: Version : 4          Header Length: 20
Tos          : 00 QOS: Routine Normal Service
Packet Length : 276           ID Number: 1543
Fragment     :                 Offset: 0
TTL          : 64             Protocol: ICMP      CheckSum: 4A5A FFFF
Source       : 9.67.22.1
Destination  : 9.67.1.2

ICMP
Type/Code    : ECHO           CheckSum: 5B3F FFFF
Id           : 4923           Seq: 11849
Echo Data    : 248

000000 B56D52DB 47A07ACA 2D523F5F 72BE75F9 |.mR.G.z.-R?_r.u.
000010 36332E6F 5A2D7969 5F7DDC7F 401715D9 |63.oZ-yi_}..@...
000020 2B9B598F 6414BB49 0D13B59F 08F8D9B9 |+Y.d..I.....
000030 099E00AF 643EE129 5C304ABF 667B4199 |....d>.)\0J.f{A.
000040 260FA3CF 4CCB6B09 3EE01BDF 6B45CD79 |&...L.k.>...kE.y
000050 33B4C2EF 2069D8E9 051FA8FF 618FFD59 |3... i.....a..Y
000060 3441DE0F 5059AAC9 2EDB721F 49215139 |4A..PY....r.I!Q9
000070 2A5B752F 5A6A60A9 7DEFF73F 15514919 |*.u/Zj`.}...?QI.
000080 0B96084F 26FB7A89 4829B85F 2B0764F9 |...0&.z.H)_.+d.
000090 7276176F 26FC7869 0945357F 1EBB24D9 |rv.o&.xi.E5...$.
0000A0 1070228F 31ECDA49 34EEEE9F 327408B9 |.p".1..I4...2t..
0000B0 5FE8A9AF 23DC2029 48C363BF 13C99099 |_...#. )H.c.....
0000C0 16342CCF 3B69CA09 1E4F14DF 59E33C79 |.4,.;i...0..Y.<y
0000D0 55972BEF 37C557E9 7D0E81FF 43788C59 |U.+7.W.}...Cx.Y
0000E0 1F46270F 36AE49C9 6C6E2B1F 34D10039 |.F'.6.I.ln+.4..9
0000F0 05659E2F 52741FA9 |.e./Rt..

IP Header      : 20      IP: 9.67.22.1, 9.67.1.2
000000 45000114 06070000 40014A5A 09431601 09430102

Data           : 256      Data Length: 256
000000 08005B3F 49232E49 B56D52DB 47A07ACA |..$......_.....: ...?I#.I.mR.G.z.
000010 2D523F5F 72BE75F9 36332E6F 5A2D7969 |...¼...9...?!... -R?_r.u.63.oZ-yi
000020 5F7DDC7F 401715D9 2B9B598F 6414BB49 |¼'." ..R..... _}..@...+Y.d..I
000030 0D13B59F 08F8D9B9 099E00AF 643EE129 |....8R.....d>.)
000040 5C304ABF 667B4199 260FA3CF 4CCB6B09 |*....#.r..t.<.,. \0J.f{A.&...L.k.
000050 3EE01BDF 6B45CD79 33B4C2EF 2069D8E9 |....,....B...QZ >...kE.y3... i..
000060 051FA8FF 618FFD59 3441DE0F 5059AAC9 |..y./.....I ....a..Y4A..PY..
000070 2EDB721F 49215139 2A5B752F 5A6A60A9 |.....$..!..-z ..r.I!Q9*.u/Zj`.
000080 7DEFF73F 15514919 0B96084F 26FB7A89 |'.7.....o.|...:i }...?QI....0&.z.
000090 4829B85F 2B0764F9 7276176F 26FC7869 |...¼...9...?.... H)_.+d.rv.o&.xi
0000A0 0945357F 1EBB24D9 1070228F 31ECDA49 |..."...R..... .E5...$.p".1..I
0000B0 34EEEE9F 327408B9 5FE8A9AF 23DC2029 |.....¼Yz.... 4...2t.._...#. )
0000C0 48C363BF 13C99099 16342CCF 3B69CA09 |.C...I.r..... H.c.....4,.;i..
0000D0 1E4F14DF 59E33C79 55972BEF 37C557E9 |.|...T...p...E.Z .0..Y.<yU.+7.W.
0000E0 7D0E81FF 43788C59 1F46270F 36AE49C9 |'.a.....I }...Cx.Y.F'.6.I.
0000F0 6C6E2B1F 34D10039 05659E2F 52741FA9 |%>...J.....z ln+.4..9.e./Rt..

```

Figure 42. A CTRACE formatted packet trace record

TCP/IP internal traces

The TCP/IP internal traces are written to the data set specified on the TCP/IP address space SYSDEBUG ddname statement. These traces provide information

on the internals of the TCP/IP address space that can be used to diagnose problems in establishing the DLC link between the TCP/IP address space and the SNALINK LU6.2 address space.

VTAM buffer traces

The VTAM buffer traces provide information on the contents of the VTAM API buffers. This information can be used to follow the data through the VTAM API interface. For details about VTAM buffer tracing and reading the trace reports, refer to *z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures*.

Finding abend and sense code documentation

The following list refers to the appropriate abend and sense code documentation for all abend and sense codes expected in the SNALINK LU6.2 network connection:

- Refer to *z/OS Communications Server: IP Messages Volume 1 (EZA)* and *z/OS Communications Server: IP and SNA Codes* for detailed SNALINK LU6.2 abend code descriptions.
- Sense codes in SNALINK LU6.2 error messages are generated by VTAM. Refer to *z/OS Communications Server: SNA Messages* and *z/OS Communications Server: IP and SNA Codes* for detailed sense code descriptions.

Finding error message documentation

The following list refers to the appropriate error message documentation for all error messages expected when using SNALINK LU6.2:

- Error messages from SNALINK LU6.2 are written to the SNALINK LU6.2 SYSPRINT data set and the MVS system console. Refer to *z/OS Communications Server: IP Messages Volume 1 (EZA)* and *z/OS Communications Server: IP and SNA Codes* for descriptions of the SNALINK LU6.2 error messages.
- Error messages from TCP/IP are written to the TCPIP SYSERROR data set. Refer to *z/OS Communications Server: IP Messages Volume 1 (EZA)* and *z/OS Communications Server: IP and SNA Codes* for descriptions of the error messages in these data sets.
- Error messages from VTAM are written to the MVS system console. Refer to *z/OS Communications Server: SNA Messages* and *z/OS Communications Server: IP and SNA Codes* for descriptions of the VTAM error messages written to the MVS system console.

Chapter 15. Diagnosing name server and dynamic domain name server (DDNS) problems

This chapter describes how to diagnose problems involving the BIND-based dynamic domain name server (DNS). Problem diagnosis involving connection optimization is also described.

Note: For additional information on diagnosing problems with a BIND-based name server, refer to *DNS and BIND, 4th Edition, Paul Albitz and Cricket Liu, 4th Edition Apr 2001 ISBN 0-596-00158-4*, available from the O'Reilly Online Catalog.

If, after reading this chapter and *DNS and BIND*, you are unable to solve a DNS-related problem and you require the services of the IBM Software Support Center, please have the following available:

- For BIND 4.9 name server, the output from syslogd and documentation from debug level 11.
- For the BIND 9 name server, the output from the debug log file with debug level of 10 or higher. You might be able to tailor the debug level to a more specific value by referring to Table 31 on page 446. See BIND 9 name server configuration logging statement for logging options. BIND 9 dynamic or static debug level may be set up to 90 for more detailed information, though it may affect server performance.

Diagnosing name server problems

The following methods are available for identifying name server problems:

- “Determining the name server version”
- “Checking messages sent to the operator’s console” on page 443
- “Checking the log messages” on page 443
- “Tools for Querying the name server” on page 445
- “Using the debug option with the name server” on page 445
- “Debugging with a resolver directive” on page 447
- “Using the remote name daemon control (rndc) program, BIND 9 name server only” on page 447
- “Using name server signals” on page 448
- “Interpreting BIND 4.9.3 name server statistics” on page 449
- “Statistics file for the Bind 9 name server” on page 450
- “Using the nsupdate command” on page 451
- “Using component trace” on page 452

These methods are discussed in the following sections.

Determining the name server version

BIND (Berkeley Internet Name Domain) is an implementation of the Domain Name System (DNS) protocols and provides an openly redistributable reference implementation of the major components of the Domain Name System, including:

- A Domain Name System server (NameD)
- A Domain Name System resolver library

- Tools for verifying the proper operation of the DNS server

The z/OS Communications Server supports two versions of BIND.

- BIND 4.9.3
- BIND 9

The name server and several name server utilities operate with either BIND 4.9.3 (v4) or BIND 9 (v9). Generally, the v4 utilities should be used with the v4 name server, and the v9 utilities should be used with the v9 name server.

You may need to determine which name server version you are communicating with (on a z/OS platform) in order to solve some problems.

Queries can be sent using dig or nslookup (version 4 or version 9) to tell if you are communicating with a BIND 4.9.3 name server or a BIND 9 name server. The queries for the name *version.bind* must be of class CHAOS and the query type must be TXT or ANY.

- If the query response is Server failed or another type of negative response, the name server that sent the answer is a BIND 4.9.3 name server.
- If the query response does not indicate failure, the name server that responded is a BIND 9 name server.

Note: The answer may contain the actual version of BIND, or it may contain other text, depending on whether the name server administrator has overridden the default text.

The following examples show typical responses that may be received when determining name server version.

- Responding name server is a BIND 9 name server:

```
nslookup - 127.0.0.1
Defaulting to nslookup version 4
Starting nslookup version 4
Default Server: localhost
Address: 127.0.0.1

> set class=chaos
> set type=any
> version.bind
Server: localhost
Address: 127.0.0.1

version.bind    text = "9.2.0"
>
```

- Responding name server is a BIND 4.9.3 name server:

```
nslookup - 127.0.0.1
Defaulting to nslookup version 4
Starting nslookup version 4
Default Server: localhost
Address: 127.0.0.1

> set class=chaos
> set type=any
> version.bind
Server: localhost
Address: 127.0.0.1

localhost can't find version.bind: Server failed
>
```

Refer to *z/OS Communications Server: IP System Administrator's Commands* for additional information.

Checking messages sent to the operator's console

Messages that display automatically on the operator's console indicate the status of your name server. Check console messages regularly to identify problems.

Messages fall into the following four categories:

- Name server initialization
- Name server initialization failure
- Name server initialization complete (such as EZZ9130I NAMED, BIND 9.2.0 IS RUNNING)
- Name server termination

For explanations of console messages, refer to *z/OS Communications Server: IP Messages Volume 4 (EZZ-SNM)*.

Checking the log messages

For the BIND 9 name server, it is important to have the syslog daemon running before the name server is started. The BIND 9 name server logging files are not initialized until after the name server configuration files have been read and processed. Therefore, any messages issued as a result of syntax or semantic errors in the BIND 9 configuration files will only appear in the syslogd output files. An MVS console message will be issued indicating that the name server ended with a fatal error.

Error messages can also be sent to a log file. You specify the name and location of this file in the syslog configuration file `/etc/syslog.conf`. Be sure to start syslogd before you start the named daemon.

For descriptions of the syslog file and the syslogd daemon, refer to the *z/OS Communications Server: IP Configuration Guide*. For information about syslog messages, refer to *z/OS Communications Server: IP Messages Volume 3 (EZY)*.

BIND 9

The following applies only to the BIND 9 name server.

- Named debug trace (up to level 99)
The debug trace can be directed to any file using the logging options in `named.conf` file. Log files are available to log important events.
- Logging can also be directed to the syslog file but severity is then limited to info and higher (does not include debug levels).
- Logging can be filtered by severity (critical, error, warning, notice, info, debug [level], dynamic).

The following is an example of the logging {} section of a `named.conf` file:

```
logging {  
    channel main_log {  
        file "/tmp/named_main.log" versions  
2 size 5M;  
        severity debug 10;  
    };  
    (blank line)  
    channel query_log {  
        file "/tmp/named_query.log" versions  
2 size 5M;
```

```

|             severity debug 10;
|             };
|         category security { query_log;
main_log; };
|         category queries { query_log;
main_log; };
|         category default { main_log; };
|         };

```

Note: This example defines 2 arbitrarily named logging channels. All debug categories are logged to the main channel so that all events can be displayed together. The security and queries categories are also sent to the query channel for faster identification of these events. The latter categories could also be pulled off the main channel altogether. Up to 30 M of disk space can be used by the 2 channels in a round robin scheme of 3 times 5 M per channel as defined.

The events are categorized and different categories can be logged to individual files if desired. The logging categories are as follows:

Category Description

default

Defines the logging options for those categories where no specific configuration has been defined.

general

Any items not otherwise categorized.

queries

Queries the server is receiving (not logged through default category).

database

Messages relating to the databases used internally by the name server to store zone and cache data.

security

Approval and denial of requests.

config

Configuration file parsing and processing.

resolver

DNS resolution, such as the recursive lookups performed on behalf of clients by a caching name server.

unmatched

Messages that **named** was unable to determine the class of or for which there was no matching **view**. A one line summary is also logged to the **client** category. This category is best sent to a file or stderr; by default it is sent to the **null** channel.

xfer-in

Zone transfers the server is receiving.

xfer-out

Zone transfers the server is sending.

notify

The NOTIFY protocol.

client

Processing of client requests.

network

Network operations.

update

Dynamic updates.

dispatch

Dispatching of incoming packets to the server modules where they are to be processed.

dnssec

DNSSEC and TSIG protocol processing.

lame-servers

Lame servers. These are misconfigurations in remote servers, discovered by BIND 9 when trying to query those servers during resolution.

Tools for Querying the name server

The `onslookup` and `NSLOOKUP` commands are helpful in diagnosing resolution of name problems in the z/OS UNIX and TSO environments, respectively. The z/OS UNIX `dig` command or the TSO `DIG` command can also be used to query name servers for problem diagnosis.

To turn on resolver tracing from `nslookup v4`, enter the following commands from the z/OS UNIX shell:

```
onslookup
set debug
```

To turn the resolver tracing off, enter the **set nodebug** command.

You can also turn on resolver tracing for `v4 nslookup` by entering the following command:

```
onslookup
set d2
```

To turn `d2` off, enter `set nod2`. For `v4 nslookup`, turning off `d2` does not turn off the resolver tracing if `debug` is also on. To turn off both `d2` and `debug`, enter `set nodebug`. With BIND 9 `nslookup`, `debug` and `d2` can be turned on and off independently. Using `debug` does not turn on resolver tracing as it does for `v4 nslookup`, but instead adds more query question and response information. Using **d2** adds some code flow traces.

For more information about the `dig`, `TSO DIG`, `onslookup` and `NSLOOKUP` commands, refer to the *z/OS Communications Server: IP System Administrator's Commands*.

Note: The `onslookup` command messages do not give a message ID for debugging and are not documented in the z/OS Communications Server library.

Using the debug option with the name server

You specify debugging in the JCL start procedure for the named server. Alternatively, you can specify debugging with the `-d` option on the `named` command or dynamically turn on debugging while the name server is running. For the BIND 4.9.3 name server, the `USR1` kill signal can be used to increment debugging to the desired level (see “Using name server signals” on page 448 for more information). For the BIND 9 name server the `rndc` command can be used to dynamically turn on tracing. Valid levels for BIND 4.9.3 for the `-d` option are in the range of 1–11, where

11 supplies the most information. Valid levels for BIND 9 are in the range of 1-99. Debugging information is sent to the file /tmp/named.run, for BIND 4.9.3. The location of the debug file for BIND 9 is specified by the logging{} statement in named.conf.

BIND 9 debugging log levels in the range of 1-99. The most useful information is contained in the messages in levels up to about 60. Specifying a level of 99 is a simple way to ensure the logging of any helpful information. High debug level logging should be utilized sparingly on high activity servers. Different logging categories may be directed to different channels, and therefore, may utilize different logging severities. Limit log size in server configuration file to avoid running out of disk space because of active and archived BIND 9 logs. With BIND 9, named.run is the default_debug logging channel, which only works if defined or default logging categories are using it *and* if the name server -d start option has been used.

The rndc command can also be helpful in dynamically changing the BIND 9 server debug level. There are rndc commands to increment the debug level, set it to the desired level, or reset the level to 0 (no debug information). It will not affect the debug level of logging files specified in channel statements in the named.conf file unless the severity level of the logging channel is dynamic. The rndc trace/notrace command affects all logging channels with a default or specified debug level of dynamic. This applies to named.run or logging channels defined in named.conf file. The default_debug logging channel, and therefore, the named.run file in the name server's working directory (specified by the directory statement in the named.conf file). See "Using the remote name daemon control (rndc) program, BIND 9 name server only" on page 447 for more information.

For BIND 9, the following types of debug information are captured at the following debug levels.

Table 31. BIND 9 debug information

BIND 9 debug level	Debugging information
1	Basic name server operation. This includes received queries, NOTIFYs from master name servers, the loading of zones, maintenance operations (including zone transfers, SOA queries by slaves, cache cleaning, and zone expirations), and task dispatching of some of the higher level functions.
2	Multicast requests.
3	Journal activity when dynamic update is enabled, DNSSEC and TSIG validation (if configured), and lower level task creation operations.
4	Incidents when a master name server has to resort to using AXFR (complete zone transfers) instead of IXFR (incremental zone transfer) because of the unavailability of journal files.
5	Captures the view being used in order to answer a request.
6	Some outgoing zone transfer requests including the query that initiates the transfer.

Table 31. BIND 9 debug information (continued)

7	The additions and deletions to journal files. and the number of bytes returned on zone transfers.
8	Most dynamic update activity and more detailed information on zone transfers.
10	Timer activity for zones.
20	Zone refresh timer updates.
90	Detailed information about task dispatching and operations.

For BIND 4.9, if `named` is started from the z/OS UNIX shell with the `-d` option, use the ampersand (&) character as a shell operator at the end of the command line to run `named` in the background. If you do not use the ampersand, the `named` tracing process occupies the z/OS UNIX shell.

With BIND 4.9, debug information generated during zone transfers is written to `/tmp/xfer.ddt.xxxxxx`, where `xxxxxx` is a unique identifier. One of these files is generated for each zone for which the `named` daemon is a secondary server. With BIND 9 zone transfer logging mostly depends on the transfer logging category which can be directed to a common or unique logging channel (file) specified by the user.

For BIND 9, `-d` does not entail working in the foreground. The latter depends on separate start options (`-f` or `-g`).

For BIND 4.9.3, if the debug level is 6 or greater, the debug information exchanged during the last initiated zone transfer is written to `/tmp/xfer.trace`.

For details on the `named` command and logging, refer to the *z/OS Communications Server: IP Configuration Reference*.

Debugging with a resolver directive

Programs that query name servers are called *resolvers*. To debug resolution of name problems, you can specify the debug option in the file `/etc/resolv.conf` (using the options `debug` directive) or in the TCP/IP configuration file. The resolver trace is sent directly into the output stream for the command using the resolver (for example, `nslookup`).

Using the remote name daemon control (rndc) program, BIND 9 name server only

The `rndc` program can be used to collect diagnostic information for BIND 9 name servers.

Configuration is required in order to use the `rndc` utility. Refer to *z/OS Communications Server: IP System Administrator's Commands* and the *z/OS Communications Server: IP Configuration Reference* for additional information.

The following `rndc` commands can be used to provide diagnostic data for the BIND 9 name server.

dumpdb

Dump the current contents of the cache (or caches if there are multiple views) into the file named by the dump-file option (by default, named_dump.db).

trace

Increment the server's debugging level by one.

trace level

Sets the server's debugging level to an explicit value.

notrace

Sets the server's debugging level to 0.

flush

Flushes the server's cache.

status

Displays status of the server.

stats

Write server statistics to the statistics file.

querylog

Toggle query logging

Refer to *z/OS Communications Server: IP System Administrator's Commands* and the *z/OS Communications Server: IP Configuration Reference* for additional information about the rndc command.

Using name server signals

You can use z/OS UNIX signals to send messages to the named daemon.

Note that the name server signals that are used to collect diagnostic information for the BIND 4.9.3 name server do not perform the same function as the signals that can be sent to the BIND 9 name server.

Note: The sigINT signal terminates the BIND 9 name server. If you send the signal to a BIND 4.9.3 name server, it dumps the name server database.

The following signals are for the BIND.4.9.3 name server:

HUP Reloads the boot file, named.boot, from the disk.

INT Dumps the contents of the name server database and hints (root server) file into the /tmp/named_dump.db file.

ABRT Dumps the current statistics of the name server in the /tmp/named.stats file.

USR1 Starts debug tracing for the name server and causes the named daemon to write debugging information to the file /tmp/named.run. USR1 can also be used to increase the debug level. Every time the USR1 signal is received, the debug level is increased until it reaches 11.

USR2 Stops debug tracing for the named daemon.

SIGWINCH

Toggles query logging on and off. Use query logging to identify resolver configuration errors. When query logging is turned on, a running name server logs every query with the syslog daemon. The syslog messages that are displayed include the IP address of the host that made the query and the query itself.

Note: Signals do not affect zone transfers in progress. If debug is on, debugging for zone transfers occurs when the command **named_xfer** is invoked.

There are three signals that currently may be used with the BIND.9 name server. Note that some of the signals may have different consequences if sent to the BIND.4.9.3 name server. The BIND.9 name server signals are:

SIGHUP

Causes the server to read **named.conf** and reload the database.

SIGTERM

Causes the server to clean up and exit.

SIGINT

Causes the server to clean up and exit.

A sample MVS start procedure is included in the **samples** directory that lets you issue these signals to the name server from the MVS operator's console. The name of the sample is **nssig**. It has one parameter, **sig**. If the sample procedure is unaltered, a typical invocation from the operator's console would be the following:

s nssig,sig=hup

Values for the **sig** parameter are the same as those for the **-s** parameter of the OMVS **kill** command. The following examples show how BIND 4.9.3 uses name server signals with the **kill** command. The process ID of the named daemon is stored in the **/etc/named.pid** file on startup.

- To dump the contents of the name server database, enter the **kill -INT \$(cat /etc/named.pid)** command from the z/OS UNIX shell, and then check the file **/tmp/named_dump.db**.
- To get short status from the named daemon, enter the **kill -ABRT \$(cat /etc/named.pid)** command from the z/OS UNIX shell, and then check the file **/tmp/named.stats**.
- To enable debug message logging for the named daemon, enter the **kill -USR1 \$(cat /etc/named.pid)** command from the z/OS UNIX shell, and then check the file **/tmp.named.run**.
- To disable debugging, enter the **kill -USR2 \$(cat /etc/named.pid)** command from the z/OS UNIX shell.
- To turn query logging on, enter the **kill -WINCH \$(cat /etc/named.pid)** command from the z/OS UNIX shell. Before logging queries, make sure that the syslog daemon is logging **LOG_INFO** messages. To turn off query logging, send another **kill -WINCH \$(cat /etc/named.pid)** signal to the name server.

Note: You can also turn query logging on by inserting the directive **options query-log** in the name server boot file or by starting the name server with **-q** on the command line.

Interpreting BIND 4.9.3 name server statistics

Message **EZZ6469I** can display name server statistics. You can use this information to get an impression of your name server's health by comparing sets of statistics over time. For example, if your name server is sending **SERVFAIL** responses, you may have a name server configuration error. Another use of statistics is to find out what a normal query load is on your name server. For example subtract the "time now" field from the "time since last boot" field to find the total number of seconds the name server has been up. Divide the total number of queries by this number to receive your queries-per-second answer.

nrrset

The number of queries that resulted in NOERROR responses with no data.

nxdomain

The number of queries that resulted in NXDOMAIN responses.

recursion

The number of queries that caused the server to perform recursion in order to find the final answer.

failure

The number of queries that resulted in a failure response other than those above.

Using the nsupdate command

The nsupdate command creates and executes Domain Name System (DNS) update operations on a host record. For nsupdate Bind 9, the -d option turns on debugging. The -d option must be specified on the nsupdate command line, as there is no interactive command to turn on debugging after nsupdate Bind 9 has been started. The -v option is used for nsupdate Bind 4 version debugging. It turns on verbose mode and displays all requests to and responses from the name server. To turn on debugging for Bind 4 version in interactive mode, enter the following commands from z/OS UNIX:

```
nsupdate
```

```
set v
```

For details on the nsupdate command, refer to the *z/OS Communications Server: IP System Administrator's Commands*.

Return codes

Following are the return codes, origination of the return codes, and explanations for the most common problems you might encounter with v4 nsupdate:

Return Code	Origin	Explanation
0	N/A	Successful.
-2	Local error	Input error.
-10	Local error	No key found in ETC\DDNS.DAT. A key is needed because either -f was specified or there is a KEY RR already in the name server data.
-11	Local error	Key in ETC\DDNS.DAT not valid. Does not authenticate the user.
-12	Local error	No response received from the name server.
-1	Local error	Represents any other (local) error not specified above.
1	Server error	Format error. The name server was unable to interpret the request.
2	Server error	Server failure. The name server was unable to process this request because of a problem with the name server.
3	Server error	Name error. The domain name specified does not exist.
4	Server error	Not implemented. The name server does not support the specified Operation code.

Return Code	Origin	Explanation
5	Server error	Refused. The name server refuses to perform the specified operation for security or policy reasons.
6	Server error	Alias error. A domain name specified in an update is an alias.
7	Server error	Name Exists error. A name already exists. This return code is only meaningful from a server in response to an ADDNAMENEW operation.
8	Server error	Record error. Indicates that a resource record (RR) does not exist. This return code is only meaningful from a server in response to a DELETE operation.
9	Server error	Zone error. Indicates that the update is to be performed on a zone for which the server is not authoritative, or that the records to be updated exist in more than one zone.
10	Server error	Ordering error. If an ordering mechanism is used (for example, a SIG RR or a SOA RR), this code indicates an ordering error. Time-signed problems are also indicated by this return code.

Using component trace

You can use the component trace function to trace data at the TCP/IP layer. In particular, the Resolver component trace may be beneficial. This information can be helpful in resolving naming problems. For detailed information on the component trace function, see Chapter 5, “TCP/IP services traces and IPCS support” on page 53.

Diagnosing problems with connection optimization

Connection optimization is a technique that uses the BIND.4.9.3 DNS for balancing IP connections and workload in a sysplex domain. You may encounter two types of problems involving connection optimization:

- Addresses not being returned
- Connection problems

Addresses not being returned

If the interface IP addresses defined for TCP/IP in the *hlq.PROFILE.TCPIP* data set and in your forward domain data file are not returned to your clients, one or more of the following situations is possible:

- The adapters associated with those addresses have not been started. If this is the problem, start the adapters.
- The adapters are started, but the stack is not registered with Workload Manager (WLM). This situation affects clients using the sysplex domain name (for example, *mvspplx.mycorp.com*, where *mvspplx* is the name of the sysplex and *mycorp.com* is the domain name). For information on how to register stacks, refer to the *z/OS Communications Server: IP Configuration Reference*.
- WLM has not refreshed the name server since the adapters associated with those addresses started. By default, WLM updates the name server every minute. If the name server has not received the most recent information from WLM, waiting at least two minutes should remedy the situation. To set the refresh, use the *-t* option on the named command.

- The name server did not start and did not return any addresses. If this is the problem, start the name server. For directions on starting the name server, refer to the *z/OS Communications Server: IP Configuration Reference*.
- The CLUSTER keyword was not coded in the primary or secondary directive in the boot file. This causes the name server to only use the statically defined names in the forward domain data file; the name server does not add dynamically generated names and optimization does not occur. If this is the problem, code the CLUSTER keyword to identify the sysplex domain.
- The host that owns the addresses defined for TCP/IP in the *hlq.PROFILE.TCPIP* data set and in the forward domain data file is short on capacity. If a host system has little or no capacity for new connections, the name server receives weights from WLM that favor other hosts. Consequently, the overloaded host system may not receive any new connections.
- No server applications are registered with WLM or they are not currently available. This affects clients that attempt to use the server application group (for example, *myserver.mvsplex.mycorp.com*, where *myserver* is the name of the server group). For information on registering servers, refer to the *z/OS Communications Server: IP Configuration Reference*.
- A server application on a particular host is not registered with WLM or is not available. This affects clients that use the group name qualified with the server name (for example, *myserver3.myserver.mvsplex.mycorp.com*).
- The sysplex connections between hosts in the sysplex are not functioning.

Connection problems

If clients attempting to reach servers in your sysplex occasionally get connection timeouts or are unable to access servers in your sysplex, one or more of the following situations are possible:

- The server running at the address given to the client application has been started, but is not totally active due to hardware problems or system definition problems. If this is the problem, refer to the *z/OS Communications Server: IP Configuration Guide*.
- The adapter associated with the address stopped recently, and that information has not yet reached the name server. Because the name server and WLM synchronize their data at one-minute intervals by default (they are not in constant communication), the name server does not learn immediately about stopped adapters. To change the length of the interval, use the named *-t* option on the named command.
- The host owning an unusable address is unreachable in your TCP/IP network. Because WLM and the name server communicate through the sysplex communication mechanisms (sysplex CTCs or XCF) and your TCP/IP network does not, it is possible that the adapter associated with the unusable address is active, but routers in the TCP/IP network cannot reach it. Avoid this type of problem by using VIPA addresses on your sysplex hosts.

Chapter 16. Diagnosing REXEC, REXECD, and RSH problems

This chapter contains diagnosis information about the classic (non-z/OS UNIX) Remote Execution Protocol (REXEC), the Remote Execution Protocol Daemon (REXECD), and the remote shell client (RSH). See “General information about REXEC and RSH” for information about REXEC and RSH and “General information about REXECD” on page 459 for information about REXECD.

General information about REXEC and RSH

REXEC and RSH are remote execution clients that allow you to execute a command on a remote host and receive the results on the local host. REXEC and RSH commands can be executed from the TSO command line or as a batch program.

Note: Refer to the *z/OS Communications Server: IP Configuration Reference* for information about defining the remote execution server.

Figure 43 shows the principle behind REXECD.

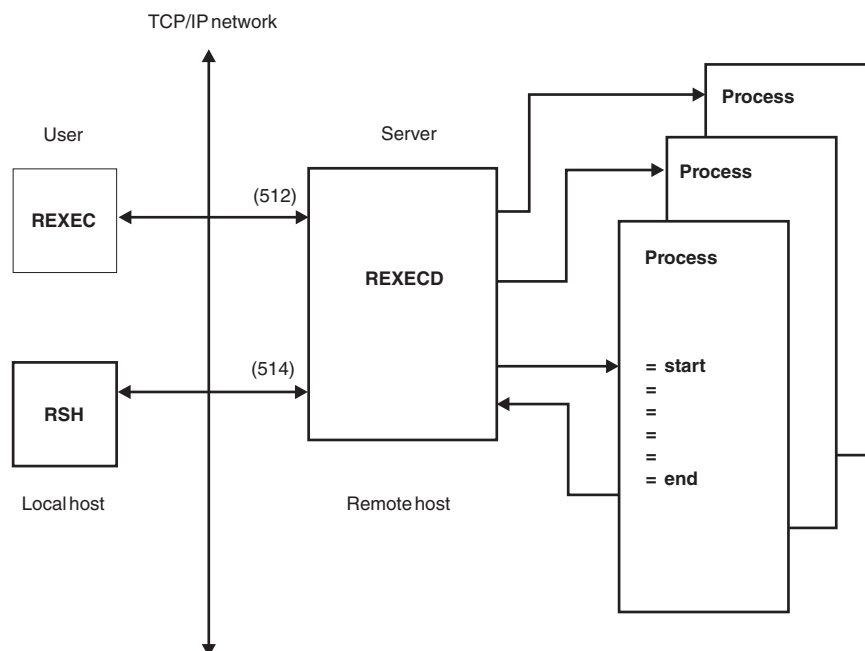


Figure 43. Remote execution protocol principle

Documentation for REXEC problem diagnosis

The following kinds of information might be required to diagnose a REXEC problem:

- REXEC console log
- REXEC debug trace

TSO console log

The TSO console log should be saved and made available, particularly if there are any error messages displayed at the console.

Activating the REXEC debug trace

To activate the REXEC debug trace, use the **REXEC -d** command.

Note: Refer to *z/OS Communications Server: IP User's Guide and Commands* for more information about REXEC commands.

REXEC trace example and explanation

Figure 44 on page 457 shows an example of an REXEC trace. Short descriptions of the numbered items in the trace follow the figure.

Note: REXEC trace output is sent to the TSO console from which the command was submitted.

```

rexec -d -l debfox -p ***** norway time
1
EZA4772I parms are -d -l debfox -p ***** norway time
EZA4756I Variables have the following assignments:
EZA4757I fhost : norway
EZA4758I userid : debfox
EZA4759I passwd : *****
2
EZA4760I command : time
EZA4801I MVS TCP/IP REXEC CS VIR4
3
EZA4780I calling GetHostResol with norway
EZA4791I Connecting to norway 197.22.190.1, port REXEC (512)
EZA4783I Passive Conn - OK on local port 1029
EZA4785I passive open complete on port 0
EZA4786I Active Conn - OK on local port 1030
EZA4788I active open complete on port 1
EZA4738I getnextnote until DD
EZA4739I Connection state changed (8681)
EZA4740I Open (8673)
EZA4774I rexec invoked;
4
EZA4775I sending: 1029 debfox ***** time
5
EZA4776I D2 len 24
EZA4778I returning from REXEC_UTIL
EZA4739I Connection state changed (8681)
EZA4740I Open (8673)
6
EZA4739I Data delivered (8682)
EZA4743I Bytes in 1
EZA4739I Data delivered (8682)
EZA4743I Bytes in 85
EZA4739I Data delivered (8682)
EZA4743I Bytes in 1
7
IKJ56650I TIME-07:42:00 PM. CPU-00:00:00 SERVICE-125 SESSION-00:00:00 FEBRUARY 6,2002
EZA4739I Connection state changed (8681)
EZA4740I Sending only (8675)
EZA4739I Connection state changed (8681)
EZA4740I Sending only (8675)
EZA4739I Connection state changed (8681)
EZA4740I Connection closing (8670)
EZA4739I Connection state changed (8681)
EZA4740I Nonexistent (8672)
EZA4739I Connection state changed (8681)
EZA4740I Connection closing (8670)
EZA4751I Returning from rcv_notices.
EZA4789I rexec complete
READY

```

Figure 44. Example of an REXEC trace

Following are short descriptions of numbered items in the trace:

- 1** The REXEC parameters that are entered at the console or received from batch.
- 2** The command that is to be sent to the remote host.
- 3** The client is attempting to resolve the name to an IP address by calling GetHostResol.
- 4** The port number, user ID, password, and command sent to the REXECD.
- 5** Length of data being sent.
- 6** Length of data passed back to the REXEC client.
- 7** Actual command output sent to the client.

RSH trace example and explanation

Figure 45 shows an example of an RSH trace. Short descriptions of numbered items in the trace follow the figure.

Note: RSH trace output is sent to the RSH console.

```
rsh -d -l USER55/MYPSWD MVS026 time
```

1

```
EZA5022I parms are RSH -d -l USER55/MYPSWD MVS026 time
EZA5006I Variables have the following assignments:
EZA5007I fhost   : MVS026
EZA5049I locuser : user1
EZA5008I userid  : USER55/MYPSWD
```

2

```
EZA5010I command : time
```

3

```
EZA5030I calling GetHostResol with MVS026
EZA5041I Connecting to MVS026 9.67.113.58, port RSH (514)
EZA5033I Passive Conn - OK on local port 1023
EZA5035I passive open complete on port 0
EZA5036I Active Conn - OK on local port 1023
EZA5038I active open complete on port 1
EZA4988I getnextnote until DD
EZA4989I Connection state changed (8681)
EZA4990I Open (8673)
EZA5046I rsh invoked;
```

4

```
EZA5025I sending: 1023 user1 USER55/MYPSWD time
```

5

```
EZA5026I D2 len 30
EZA5047I returning from RSH_UTIL
EZA4989I Connection state changed (8681)
EZA4990I Open (8673)
EZA4989I Data delivered (8682)
```

6

```
EZA4993I Bytes in 1
EZA4989I Data delivered (8682)
EZA4993I Bytes in 78
EZA4989I Data delivered (8682)
EZA4993I Bytes in 1
```

7

```
TIME-05:42:59 PM. CPU-00:00:00 SERVICE-12150 SESSION-00:00:02 FEBRUARY 19,2001
EZA4989I Connection state changed (8681)
EZA4990I Sending only (8675)
EZA4989I Connection state changed (8681)
EZA4990I Sending only (8675)
EZA4989I Connection state changed (8681)
EZA4990I Connection closing (8670)
EZA4989I Connection state changed (8681)
EZA4990I Nonexistent (8672)
EZA4989I Connection state changed (8681)
EZA4990I Connection closing (8670)
EZA5001I Returning from recv_notices.
EZA5048I rsh complete
```

Figure 45. Example of an RSH trace

Following are short descriptions of numbered items in the trace:

- 1** The RSH parameters that are entered at the console or received from batch.
- 2** The command that is sent to the remote host.
- 3** The client is attempting to resolve the name to an IP address by calling GetHostResol.
- 4** The port number, user ID, password, and command sent to the REXECD.
- 5** Length of data being sent.
- 6** Length of data passed back to the RSH client.
- 7** Actual command output sent to the client.

General information about REXECD

The remote execution server allows execution of a TSO batch command that has been received from a remote host. REXECD supports both the remote execution command (REXEC) and remote shell (RSH) client protocols.

Documentation for REXECD problem diagnosis

The following kinds of information might be required to diagnose a REXECD problem:

- REXECD console log
- REXECD traces

MVS system console log

The MVS system console log should be saved and made available, particularly if there are any error messages displayed at the console.

Starting REXECD server traces

To run the REXECD trace, REXECD must be started with one or more of the following options on the TRACE parameter in the PROC statement:

LOG

Specifies to write trace records to the SYSPRINT data set.

SEND

Specifies to send trace records to the REXEC or RSH client.

CLIENT

Specifies a specific client host for which trace records are to be produced.

ALLCLIENTS

Specifies that host records are to be produced for all clients.

Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about the options. Refer to the *z/OS MVS JCL Reference* for information about the length limit of the PARM= parameter on the exec statement in the start procedure.

Notes:

1. REXECD trace output is included in the job output log.
2. If more than one trace option is selected, the options must be enclosed within parentheses.

Example of an REXECD trace of a client using the SEND command

Figure 46 shows a portion of an example of an REXECD trace of a client using a SEND command. Short descriptions of numbered items in the trace follow the figure.

```
EZA4801I MVS TCP/IP REXEC CS V1R4
1
EZA4383I SSCSARAY:0: JOB00043 40
    EZA4383I SSCSARAY:0: JOB00043 80 2
EZA4383I SSCSARAY:0: JOB00043 80
3
EZA4383I SSCSARAY:0: JOB00043 20
    EZA4385I SSSORT(CTRL): 00000000
4
EZA4392I S99ret: 00000000, A RSHD NEWALTON.RSHD5.JOB00043.D0000105.?
5
TIME-12:04:12 PM. CPU-00:00:00 SERVICE-1157 SESSION-00:00:01 MARCH 9,1998
    EZA4393I S99ret: 00000000
6
EZA4390I SSSORT(next): 00000004
```

Figure 46. Example of an REXECD trace of a client using a SEND command

Following are short descriptions of numbered items in the trace:

- 1** JOB00043 is the JES job number. The 40 indicates the job is waiting for execution.
- 2** The 80 indicates the job is currently active.
- 3** The 20 indicates the job is on the output queue.

Note: Refer to *z/OS Communications Server: IP Messages Volume 1 (EZA)* for more information about the individual messages in the trace.

- 4** This line shows the return code from the dynamic allocation of the JES data sent back to the client.
- 5** Actual command output sent to the client.
- 6** This is the return code expected when there is no more work to do.

Example trace of an RSH client using the SEND command

Figure 47 on page 461 shows a portion of an example of a trace of an RSH client using a SEND command. Short descriptions of numbered items in the trace follow the figure.

```

1
EZA4383I SSCSARAY:0: JOB00043 20
EZA4385I SSSORT(CTRL): 00000000
EZA4389I SSSORT(init): 00000004
2
EZA4383I SSCSARAY:1: JOB00044 40
EZA4383I SSCSARAY:0: JOB00043 20
EZA4385I SSSORT(CTRL): 00000000
EZA4389I SSSORT(init): 00000004
3
EZA4383I SSCSARAY:1: JOB00044 80
EZA4383I SSCSARAY:0: JOB00043 20
EZA4385I SSSORT(CTRL): 00000000
EZA4389I SSSORT(init): 00000004
4
EZA4383I SSCSARAY:1: JOB00044 20
EZA4385I SSSORT(CTRL): 00000000
5
EZA4392I S99ret: 00000000, A RSHD NEWALTON.RSHD5.JOB00044.D0000105.?
6
TIME-12:07:02 PM. CPU-00:00:00 SERVICE-1134 SESSION-00:00:00 MARCH 9,1998
EZA4393I S99ret: 00000000
7
EZA4390I SSSORT(next): 00000004

```

Figure 47. Example of a trace of an RSH client using a SEND command

Following are short descriptions of numbered items in the trace:

- 1** JOB00043 is a previous job that has completed.
- 2** The 40 indicates that job JOB00044 is waiting for execution.
- 3** The 80 indicates that job JOB00044 is currently active.
- 4** The 20 indicates that job JOB00044 is on the output queue.

Note: Refer to *z/OS Communications Server: IP Messages Volume 1 (EZA)* for more information about the individual messages in the trace.

- 5** This line shows the return code from the dynamic allocation of the JES data sent back to the client.
- 6** Actual command output sent to the client
- 7** This is the return code expected when there is no more work to do.

Chapter 17. Diagnosing z/OS UNIX REXEC, REXECD, and RSHD problems

This chapter contains diagnosis information about the z/OS UNIX remote execution protocol (REXEC), the remote execution protocol daemon client (REXECD), and the remote shell daemon (RSHD).

Setting up the inetd configuration file

The inetd program is a generic listener program used by such servers as z/OS UNIX TELNETD and z/OS UNIX REXECD. Other servers such as z/OS UNIX FTPD have their own listener program and do not use inetd.

The inetd.conf file is an example of the user's configuration file. It is stored in the /etc directory. Upon startup, the servers for z/OS UNIX TELNETD, rshell, rlogin, and rexec are initiated if they have been defined in /etc/inetd.conf. If it does not include z/OS UNIX TCP/IP applications, add the information shown in Figure 48:

#=====						
# service	socket	protocol	wait/	user	server	server program
# name	type		nowait		program	arguments
#=====						
#						
shell	stream	tcp	nowait	OMVSKERN	/usr/sbin/orshd	rshd -l
exec	stream	tcp	nowait	OMVSKERN	/usr/sbin/orexecd	rexecd -LV
oteln	stream	tcp	nowait	OMVSKERN	/usr/sbin/oteln	oteln
login	stream	tcp	nowait	bpxroot	/bin/rlogin	rlogin -d 1

Figure 48. Adding applications to /etc/inetd.conf

Note: For IPv6 support, specify tcp6 for the protocol.

To establish a relationship between the servers defined in the /etc/inetd.conf file and specific port numbers in the z/OS UNIX environment, ensure that statements have been added to ETC.SERVICES for each of these servers. See the sample ETC.SERVICES installed in the /usr/lpp/tcpip/samples/services directory for how to specify ETC.SERVICES statements for these servers.

Note: It is important that the service name in /etc/inetd.conf (login in 1) matches the service name in /etc/services
login 513/tcp

The traces for both the z/OS UNIX REXECD server and the z/OS UNIX RSHD server are enabled by options in the inetd configuration file (/etc/inetd.conf). See Figure 49 on page 464.

```

#=====
# service | socket | protocol | wait/ | user | server | server program
# name    | type  |         | nowait|      | program| arguments
#=====
#
shell    stream  tcp      nowait OMVSKERN /usr/sbin/orshd rshd -d 2
exec     stream  tcp      nowait OMVSKERN /usr/sbin/orexecd rexecd -d 3

```

Figure 49. Setting traces in /etc/inetd.conf

The traces are turned on for both servers by passing a -d argument to the server programs. **2** is the RSHD server and **3** is the REXECD server. All commands executed after the debug flags have been turned on in the inetd configuration file and after the inetd server has reread the file will produce trace output.

The trace is written in formatted form to the syslogd facility name daemon with a priority of debug. The trace data can be routed to a file in your Hierarchical File System by specifying the following definition in your syslogd configuration file (/etc/syslogd.conf):

```

#
# All ftp, rexecd, rshd
# debug messages (and above
# priority messages) go
# to server.debug.a
#
daemon.debug                /tmp/syslogd/server.debug.a

```

In this example, the trace data is written to /tmp/syslogd/daemon.debug.a in your hierarchical file system. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about syslogd.

For more information about inetd, refer to *z/OS UNIX System Services Planning* or *Accessing OpenEdition MVS from the Internet*.

Diagnosing z/OS UNIX REXEC

The following kinds of information can help you diagnose a z/OS UNIX REXEC problem:

- A message beginning with EZYRC
- A code
- An z/OS UNIX REXEC debug trace
- A REXECD debug trace from the foreign host

Activating the z/OS UNIX REXEC debug trace

To activate the z/OS UNIX REXEC debug trace, specify the -d option.

z/OS UNIX REXEC trace example and explanation

The z/OS UNIX REXEC can be invoked using either rexec or orexec. Enter one of the following commands with either an IP address or a host name.

IPv4

```
orexec -d -l debfox -p mypwd -s 1512 fec0:0:0:12BE::1 ls -al
```

IPv6

```
orexec -d -l debfox -p mypwd -s 1512 197.22.190.1 ls -al
```

The following are examples of the trace output:

IPv4

EZYRC02I Host: 197.22.190.1, user debfox, cmd ls -al, port 1512

IPv6

EZYRC02I Host: fec0:0:0:12BE::1, user debfox, cmd ls -al, port 1512

EZYRC01I Calling function rexec_af with the following:

EZYRC02I Host: fec0:0:0:12BE::1, user debfox, cmd ls -al, port 1512

EZYRC01I shows that the z/OS UNIX REXEC function has been called in the run-time libraries. EZYRC02I shows the parameters that have been passed to the REXEC() function in the run-time library. EZYRC19I shows the socket descriptor being used for the data connection and the control (or standard error) connection.

Diagnosing z/OS UNIX REXECD

The following kinds of information can help you diagnose a z/OS UNIX REXECD problem:

- A message beginning with EZYRD
- A code
- A z/OS UNIX REXECD debug trace
- A trace from the z/OS UNIX REXECD client

Activating the z/OS UNIX REXECD debug trace

The z/OS UNIX REXEC can be invoked using either rexecd or orexecd. To activate the z/OS UNIX REXECD debug trace, specify the -d option in the /etc/inetd.conf file.

z/OS UNIX REXECD trace example and explanation

These examples are in the file specified in syslogd.conf.

Note: Ensure syslogd is running before collecting these traces and that the file has been properly specified.

```
Jun 12 13:31:47 rexecd.851981.: EZYRD31I MVS OE REXECD BASE
```

The entry is stamped with the date, time, the name of the daemon and the order number of the daemon, the message number (EZAYRD31I), and related information, as shown in the following example.

```
Jun 12 13:31:49 rexecd.851981.: EZYRD03I Remote address = 9.67.113.61
Jun 12 13:31:49 rexecd.851981.: EZYRD05I clisecport = 1029
Jun 12 13:31:49 rexecd.851981.: EZYRD08I User is: user21
Jun 12 13:31:49 rexecd.851981.: EZYRD09I Command is: ls -l
Jun 12 13:31:49 rexecd.851981.: EZYRD12I Name is: USER21, user is user21
Jun 12 13:31:49 rexecd.851981.: EZYRD13I dir is: /u/user21
Jun 12 13:31:49 rexecd.851981.: EZYRD14I uid is: 21, gid is 0
```

For an explanation of the messages, refer to *z/OS Communications Server: IP Messages Volume 1 (EZA)*.

Diagnosing z/OS UNIX RSHD

The following kinds of information can help you diagnose a z/OS UNIX RSHD problem:

- A message beginning with EZYRS
- A code
- A z/OS UNIX RSHD debug trace
- A trace from the RSH client

Activating the z/OS UNIX RSHD debug trace

The z/OS UNIX RSHD can be invoked using either `rshd` or `orshd`. To activate the z/OS UNIX RSHD debug trace, specify the `-d` option in the `/etc/inetd.conf` file.

z/OS UNIX RSHD trace example and explanation

These examples are from the file specified in `syslogd.conf`.

Note: Ensure `syslogd` is running before collecting these traces and that the file exists and has been properly specified.

```
Jun  9 12:10:04  rshd.4653080.: EZYRS01I  MVS OE RSHD BASE
```

The entry is stamped with the date, time, name of daemon and the order number of the daemon, the message number (EZYRS01I), and related information, as shown in the following example.

```
Jun  9 12:10:06  rshd.4653080.: EZYRS12I  Clisecport = 1020
Jun  9 12:10:06  rshd.4653080.: EZYRS21I  Remote user is: OS2USER
Jun  9 12:10:06  rshd.4653080.: EZYRS22I  Local user is: user21
Jun  9 12:10:06  rshd.4653080.: EZYRS23I  Command is: ls -l
```

For an explanation of the messages, refer to *z/OS Communications Server: IP Messages Volume 3 (EZY)*.

If the `-A` option is specified in `/etc/inetd.conf`, the z/OS UNIX RSHD server will not execute a command when the client host IP address cannot be resolved to a host name.

Resolving garbage errors

There are a few situations where the z/OS UNIX RSHD server may encounter an error so early in the processing of a command that the server has not yet established a proper EBCDIC-to-ASCII translation. In such a situation, the client end user may see garbage data returned to his or her terminal. A packet trace will reveal that the response is in fact returned in EBCDIC, which is the reason for the garbage look on an ASCII workstation. This can happen if the z/OS UNIX name resolution has not been configured correctly, so the z/OS UNIX RSHD server, for example, was not able to resolve IP addresses and host names correctly. If your RSH clients encounter such a problem, go back and check your name resolution setup. If you are using a local hosts table, make sure that the syntax of the entries in your hosts file is correct.

Chapter 18. Diagnosing network database system (NDB) problems

The network database system (NDB) allows workstation or mainframe users to issue SQL statements interactively, or to invoke NDB services from within a C application program. NDB services can then be used to pass SQL statements to the DB2 subsystem and handle replies from the DB2 subsystem. The NDB client uses the remote procedure call (RPC) to package the request and issue a remote procedure call that sends the request to the NDB server. The NDB server passes the SQL request to the DB2 subsystem for processing. When processing is complete, the DB2 subsystem passes data or a return code, or data and a return code, to the NDB server, which returns them to the NDB client.

Notes:

1. Refer to *z/OS Communications Server: IP Messages Volume 1 (EZA)* for more information about NDB usage.
2. Unless specifically shown otherwise, all examples in this chapter are valid for V1R2.

Note:

The components of the Network Database System are shown in Figure 50.

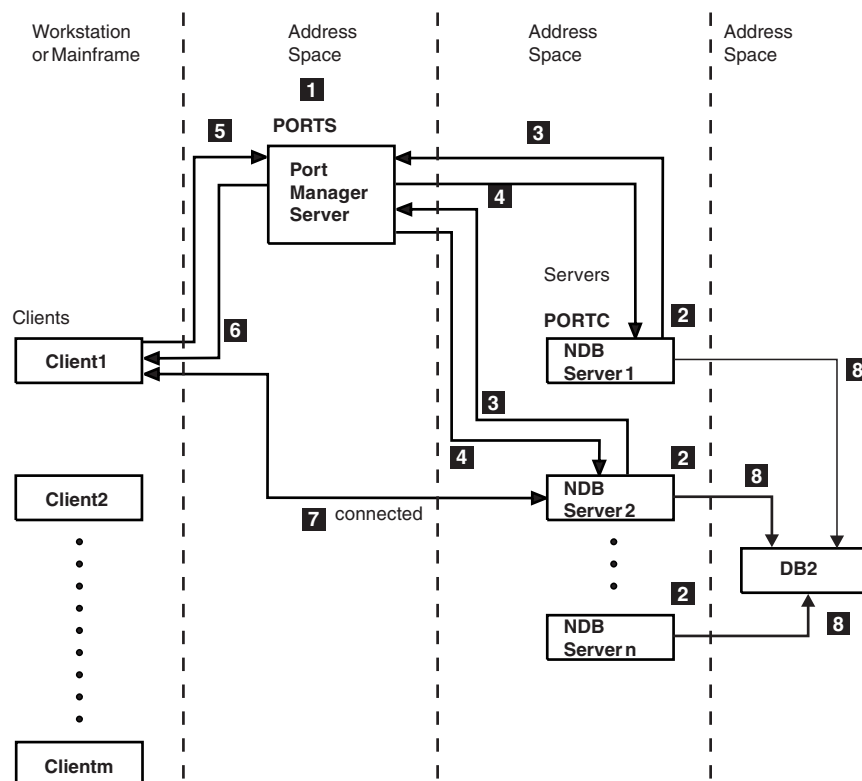


Figure 50. Components of the network database system

Following are a list of steps corresponding to the numbered items in the figure:

1. Bring up the NDB port manager (PORTS). When PORTS is started, it

registers its program number with the portmapper, so that portmapper knows on which port PORTS is listening.

- 2** Bring up the NDB servers (to a maximum of 20). Note that the C multitasking facility is used by PORTC. The number of NDB servers brought up is specified as a startup parameter.
- 3** Each NDB server issues a request to PORTS for a program number, through the NDB port client (PORTC).
- 4** PORTS updates its port status and returns a program number.
- 5** When an NDB client wants NDB services, it calls PORTS at its program number and requests the program number of an available NDB server.
- 6** PORTS returns the program number of an available NDB server.
- 7** The NDB client then calls the NDB server with the program number and RPC looks up the port number that is used for the connection.
- 8** With the connection established, the client can use NDB services to issue SQL requests by using the NDBCLNT command.

Multiple PORTC PROCs can be started, each supporting 1–20 NDB servers. Each PORTC address space can access a different DB2 subsystem. A total of 100 NDB servers, across from 5–100 PORTC address spaces, is supported.

Note: Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about setting up and starting the NDB clients, the NDB port manager, the NDB servers, and the NDB port client.

Documentation for NDB problem diagnosis

The following kinds of information are always required to diagnose an NDB problem:

- Environment description
 - Client environment (for example, OS/2, AIX, or MVS), client level of TCP/IP, and current CSD level for workstation environments
 - Host level of TCP/IP and current maintenance level
- Console output
 - Console output from the NDB server (PORTC)
 - Keystrokes entered, in sequence, from the client side
 - All error messages

The following trace is requested if needed:

- DB2 trace

Definitions

The following definitions are required for you to use NDB:

- The DB2 subsystem that you intend to use with NDB must be defined.
- Portmapper must be installed and functional.
- The NDB port manager address space must be started.

The NDB port manager address space consists of one module, PORTMGR.
- The NDB port client and server address spaces must be started.
 - The NDB server address space consists of two modules, PORTCLNT (the NDB port client) and NDBSS (the NDB server). The NDB server code uses

the C multitasking facility and can manage from 1–20 NDB servers within this address space. Refer to the *z/OS Communications Server: IP Configuration Reference* for information about configuring and starting the address spaces.

- For all platforms, except MVS, the NDB client code must be moved to the platform from which the user plans to issue SQL statements, and an executable file must be built. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information.

Diagnosing NDB problems

Most of the information you need can be gathered through return codes.

Follow these steps to gather information you need:

1. Check the return code and error message. Refer to the *z/OS Communications Server: IP User's Guide and Commands* and *z/OS Communications Server: IP Messages Volume 1 (EZA)* for more information about the return codes.
2. If the return code is +1 or -20000, make sure that the Portmapper is up and running (use RPCINFO), that the path to the host running the DB2 subsystem is available (use PING), and that the DB2 subsystem is up and running (check the MVS system console). Any of these conditions could result in an RPC error or timeout.
3. If the return code is -20100, an incompatibility exists between the NDB client settings of, and the NDB server accepted values for, specific fields of the NDBC control block. Currently accepted settings are given in the return code explanations in the *z/OS Communications Server: IP User's Guide and Commands*. Another possible cause of the problem could be corruption of NDBC control block on either the client or server side. If you believe this is the problem, contact the IBM Software Support Center.

If there are problems obtaining DB2 data from the database, use SPUFI to check the system tables by performing the following analysis steps:

1. Has a BIND has been issued for DBRM DBUTIL2?

To verify a BIND has been issued for DBRM DBUTIL2, issue the following SQL query:

```
select * from sysibm.sysplan where name='EZAND320'
```

If the plan is not found, refer to the *z/OS Communications Server: IP Configuration Reference* for information about binding the DBRM DBUTIL2 (NDBSETUP) to create the plan EZAND320.

2. Is the TSO user ID that is trying to use the plan EZAND320 authorized?

To verify that the TSO user that is trying to use NDB (trying to execute the plan EZAND320) is authorized, issue the following SQL query:

```
select * from sysibm.sysplanauth where name='EZAND320'
```

If neither the user ID executing the plan nor PUBLIC is authorized (that is, listed in the table under the column grantee), execute one of the following commands:

- Grant execute on plan EZAND320 to *user_id*
- Grant execute on plan EZAND320 to public

The *user_id* is the TSO user ID that will execute EZAND320.

3. Has the procedure PORTC been updated to point to the correct DB2 load library with the suffix DSNLOAD?

Verify that the level of DB2 being used is CS for OS/390 V2R3, or higher. Check the PORTC PROC to ensure that it is pointing to the same subsystem that was specified in the PORTC start up parameter DB2SSID, and that the BIND for DBUTIL2 was completed.

NDB trace examples and explanations

Figure 51 on page 471 shows an example of a trace of the NDB port manager showing the console trace when two NDB servers are started and one NDB client is invoked. It corresponds to the NDB port client and server trace found in Figure 52 on page 473.

Notes:

1. The NDB port manager tracing is off by default. To turn it on, the IBM Software Support Center must build a module using the DEF(DEBUG) option and send it to the customer.
2. NDB trace output is included in the job log output from the started NDB procedure.


```

1
18:58:42 EZA3950I NDB PORT MANAGER FOR CS/390 V2R10 STARTED

2
NDBPS received request. Calling PORTMGR.
  Entering program PORTMGR. SSCB at entry is:
  who is 1. smid is MVSL. suid is SYSADM.
  prognum is 0. portnum is 0. status is 1.
  Entering case: who is NDBSRV(1)
  Entering case: status is NEW(1) or INIT(0)
  Available NDB Server found. Prognum is 536870944
  Exiting program PORTMGR. SSCB at exit is:
  who is 1. smid is MVSL. suid is SYSADM.
  prognum is 536870944. portnum is 0. status is 3.

2
NDBPS received request. Calling PORTMGR.
  Entering program PORTMGR. SSCB at entry is:
  who is 1. smid is MVSL. suid is SYSADM.
  prognum is 0. portnum is 0. status is 1.
  Entering case: who is NDBSRV(1)
  Entering case: status is NEW(1) or INIT(0)
  Available NDB Server found. Prognum is 536870945
  Exiting program PORTMGR. SSCB at exit is:
  who is 1. smid is MVSL. suid is SYSADM.
  prognum is 536870945. portnum is 0. status is 3.

3
NDBPS received request. Calling PORTMGR.
  Entering program PORTMGR. SSCB at entry is:
  who is 2. smid is . suid is .
  prognum is 0. portnum is 0. status is 1.
  Entering case: who is NDBCLNT(2)
  Entering case: status is NEW(1)
  Found PORTINFO entry with STATUS of NOT_BUSY. Updating WHO's SSCB fields from PORTINFO entry.
  Exiting program PORTMGR. SSCB at exit is:
  who is 2. smid is MVSL. suid is SYSADM.
  prognum is 536870944. portnum is 0. status is 2.

4
NDBPS received request. Calling PORTMGR.
  Entering program PORTMGR. SSCB at entry is:
  who is 2. smid is . suid is .
  prognum is 536870944. portnum is 0. status is 5.
  Entering case: who is NDBCLNT(2)
  Entering case: status is DONE(5)
  Found PORTINFO entry with PROGNUM same as WHO's SSCB PROGNUM.
  Setting STATUS in both to NOT_BUSY and reinitializing other fields of PORTINFO.
  Exiting program PORTMGR. SSCB at exit is:
  who is 2. smid is . suid is .
  prognum is 536870944. portnum is 0. status is 3.

```

Figure 51. NDB port manager trace with two NDB servers started and one client invoked

Following are short descriptions of the numbered items in the trace:

- 1** This message indicates that the NDB port manager procedure has successfully completed startup.
- 2** The following 10 messages indicate that one NDB server has been started. These 10 messages are printed out once for each NDB server started, but with each NDB server being assigned a unique program number.
- 3** The following 10 messages are issued each time an NDB client contacts the NDB port manager for an available NDB server. This is done when an NDB client is first invoked.
- 4** The following 11 messages are issued each time an NDB client-user issues

the NDB END command. The END command indicates to the NDB port manager that this NDB client session has finished and the NDB server associated with it is again available.

Figure 52 on page 473 shows a trace of the NDB port client and NDB servers when two NDB servers are started, and one NDB client is invoked. It corresponds to the NBD port manager trace shown in Figure 51 on page 471.

Notes:

1. NDB port client and NDB server tracing is off by default. It can be turned on by specifying the TRACE parameter at PORTC startup with the option ON or YES.
2. NDB trace output is included in the job log output from the started NDB procedure.

```

1
18:59:07 EZA4000I PORTCLNT ENTRY FOR MVS VERSION 3
2
Tracing is now active. Enjoy your output!
3
Program PORTCLNT being executed.
  The input parms from startup are as follows:
  7 parms were supplied.
  argv(0), hopefully name of this module, is PORTCLNT
  argv(1), hopefully host name, is MVSL
  argv(2), hopefully userid to run under, is SYSADM
  argv(3), hopefully constant, is NDBSRV
  argv(4), hopefully DB2 subsystem name, is D23
  argv(5), hopefully number of servers to start, is 2
  argv(6), hopefully trace on indicator, is on
4
About to call clnt_create
  Returned from clnt_create without error
5
Timeout value is 300
6
PORTCLNT invoked with Requester NDBSRV
  SSCB of PCb contents after setup:
  WHO is: 1
  SMID is: MVSL
  SUID is: SYSADM
  PROGNUM is: 0(Dec)
  PORTNUM is: 0(Dec)
  STATUS is: 1
7
PORTCLNT: DB2 name is 3 chars long.
  Copied DB2 name into db2sys, D23
8
18:59:08 EZA4007I NUMBER OF NDB SERVERS BEING STARTED IS 2
9
tinit of MTF about to be called
  tinit of MTF successfully called
10
Server number 1 is starting up
  MVS only code: about to call NDB Port Manager
  Successfully returned from call NDB Port Manager
  SSCB of Result contents after ports_msg_1:
  WHO is: 1
  SMID is: MVSL
  SUID is: SYSADM
  PROGNUM is: 536870944(Dec)
  PORTNUM is: 0(Dec)
  STATUS is: 3
11
18:59:11 EZA4011I SERVER 1 STARTED. PROGNUM IS 20000020(HEX), 536870944(DEC).
12
tsched of MTF about to be called
  Parms being passed are:
  result->prognum is 536870944
  db2sys is D23
  trace is 1

```

Figure 52. NDB port client trace with two NDB servers started and one client invoked (Part 1 of 7)

```

13
Server number 2 is starting up
  MVS only code: about to call NDB Port Manager
  Successfully returned from call NDB Port Manager
  SSCB of Result contents after ports_msg_1:
  WHO is: 1
  SMID is: MVSL
  SUID is: SYSADM
  PROGNUM is: 536870945(Dec)
  PORTNUM is: 0(Dec)
  STATUS is: 3
14 18:59:11 EZA4011I SERVER 2 STARTED. PROGNUM
  IS 20000021(HEX), 536870945(DEC).
15 tsched of MTF about to be called     Parms being passed are:      result->prognum is 536870945
  db2sys is D23      trace is 1
16 18:59:13 EZA4150I NDB SERVER STARTED WITH PROGNUM 20000020(HEX), 536870944(DEC)
17 Got DB2 name into NDBSS. It is: D23      Now have copied it into db2ssid. It is: D23
  Value of Trace global variable is 1
18 NDBSRV about to be called on behalf of NDB Client      18:59:36 EZA4151I MVS NDB SERVER
  RECEIVED A CALL FROM HOST USERID user1
19 Entering program NDBSRV
  Static var NewUser is 1
NDBC contents at NDBSRV entry is:
-----
ndbrel   is 1
ndbver   is 2
ndbcb    is NDBC
ndbsrc   is 0
ndbappl  is 1
ndbstat  is 0
ndbsname is netdbsrv
ndbusrid is user1
ndbpswd  is not echoed in trace
ndbrqdl  is 77
ndbrqd   is 3bb4668 (Hex)
ndbrpdln is 8192
ndbrpd   is 3bc9ff8 (Hex)
ndbrqd contents is:
create table empinfo (empno int, name char(15),salary dec(8,2),hiredate date)
20 NDBC Reply buffer has been initialized
  NDBC Host userid and password verified
  NDBC Control Block header fields verified
21 Entered NewUser conditional code
Calling DBOpen from NDBSRV. name is D23
  Entering DBOpen function
  DBUTIL2: ssid is D23 and plan is DBUTIL2
  DB OPEN: rtc is 0 and rsc is 0
  Exiting DBOpen function
  In NDBSRV:Open: rtc is 0. rsc is 0(Hex).
  CAF OPEN DB was successful.
  End of NewUser conditional code. NewUser is 0.
22 Processing SQL statement. Calling SQLOpen.
  Entering SQLOpen function
  Value of Init_Done is 0
  Value of LocaTStat is 0
  Value of rowBuffer is 0
  rowBufferp is set at 0
  Value of colBytes is 0
  Value of numEntries is 0
  Value of numBytes is 0

```

Figure 52. NDB port client trace with two NDB servers started and one client invoked (Part 2 of 7)

```

23 In SQLOpen, in conditional code for not Init_Done
    End of not Init_Done conditional code
    Value of Init_Done is 1
    Value of numEntries is 60
    Value of numBytes is 2656
24 SQL variables set up. SQLLEN is 77
    and SQLSTR is <create table empinfo (empno int, name char(15),salary dec(8,2),hiredate date)>
    token, representing type of SQL stmt, is 7
25 Exiting SQLOpen function
    Value of Init_Done is 1
    Value of LocalStat is 0
    Value of rowBuffer is 0
    rowBufferp is set at 0
    Value of colBytes is 0
    Value of numEntries is 60
    Value of numBytes is 2656
    Back from SQLOpen. RC is 0. NDBSRC is 0
26 Exiting program NDBSRV
    Static var NewUser is 0
    NDBC contents at NDBSRV exit is:
    -----
    ndbre1   is 1
    ndbver   is 2
    ndbcb    is NDBC
    ndbsrc   is 0
    ndbappl  is 1
    ndbstat  is 0
    ndbsname is netdbsrv
    ndbusrid is USER1
    ndbpswd  is not echoed in trace
    ndbrqdln is 77
    ndbrqd   is 3bb4668 (Hex)
    ndbrpdln is 8192
    ndbrpd   is 3bc9ff8 (Hex)
27 NDBSRV about to be called on behalf of NDB Client
    18:59:43 EZA4151I MVS NDB SERVER RECEIVED A CALL FROM HOST USERID user1
19 Entering program NDBSRV
    Static var NewUser is 0
    NDBC contents at NDBSRV entry is:
    -----
    ndbre1   is 1
    ndbver   is 2
    ndbcb    is NDBC
    ndbsrc   is 0
    ndbappl  is 1
    ndbstat  is 0
    ndbsname is netdbsrv
    ndbusrid is user1
    ndbpswd  is not echoed in trace
    ndbrqdln is 69
    ndbrqd   is 3bcbf60 (Hex)
    ndbrpdln is 8192
    ndbrpd   is 3bd7ff8 (Hex)
    ndbrqd contents is:
    insert into empinfo values (10001, 'Andersen', 23456.78, '01/02/1983')

```

Figure 52. NDB port client trace with two NDB servers started and one client invoked (Part 3 of 7)

```

20 NDBC Reply buffer has been initialized
   NDBC Host userid and password verified
   NDBC Control Block header fields verified
22 Processing SQL statement. Calling SQLOpen.
   Entering SQLOpen function
   Value of Init_Done is 1
   Value of LocalStat is 0
   Value of rowBuffer is 0
   rowBufferp is set at 0
   Value of colBytes is 0
   Value of numEntries is 60
   Value of numBytes is 2656
28 In SQLOpen, in else code, therefore Init_Done
   End of Init_Done code
   Value of Init_Done is 1
   Value of numEntries is 60
   Value of numBytes is 2656
24 SQL variables set up. SQLLEN is 69
   and SQLSTR is <insert into empinfo values (10001, 'Andersen', 23456.78,'01/02/1983')>
   token, representing type of SQL stmt, is 6
25 Exiting SQLOpen function
   Value of Init_Done is 1
   Value of LocalStat is 0
   Value of rowBuffer is 0
   rowBufferp is set at 0
   Value of colBytes is 0
   Value of numEntries is 60
   Value of numBytes is 2656
   Back from SQLOpen. RC is 0. NDBSRC is 0
26 Exiting program NDBSRV
   Static var NewUser is 0
   NDBC contents at NDBSRV exit is:
   -----
ndbre1   is 1
ndbver   is 2
ndbcb    is NDBC
ndbsrc   is 0
ndbappl  is 1
ndbstat  is 0
ndbsname is netdbsrv
ndbusrid is USER1
ndbpswd  is not echoed in trace
ndbrqdln is 69
ndbrqd   is 3bcbf60 (Hex)
ndbrpdln is 1
ndbrpd   is 3bd7ff8 (Hex)

```

Figure 52. NDB port client trace with two NDB servers started and one client invoked (Part 4 of 7)

```

27 NDBSRV about to be called on behalf of NDB Client
   18:59:44 EZA4151I MVS NDB SERVER RECEIVED A CALL FROM HOST USERID user1
19 Entering program NDBSRV
   Static var NewUser is 0
   NDBC contents at NDBSRV entry is:
   -----
   ndbre1   is 1
   ndbver   is 2
   ndbcb    is NDBC
   ndbsrc   is 0
   ndbappl  is 1
   ndbstat  is 0
   ndbsname is netdbsrv
   ndbusrid is user1
   ndbpswd  is not echoed in trace
   ndbrqdl  is 21
   ndbrqd   is 3bcbf90 (Hex)
   ndbrpdln is 8192
   ndbrpd   is 3bd7ff8 (Hex)
   ndbrqd contents is:
   select * from empinfo
20 NDBC Reply buffer has been initialized
   NDBC Host userid and password verified
   NDBC Control Block header fields verified
22 Processing SQL statement. Calling SQLOpen.
   Entering SQLOpen function
   Value of Init_Done is 1
   Value of LocalStat is 0
   Value of rowBuffer is 0
   rowBufferp is set at 0
   Value of colBytes is 0
   Value of numEntries is 60
   Value of numBytes is 2656
28 In SQLOpen, in else code, therefore Init_Done
   End of Init_Done code
   Value of Init_Done is 1
   Value of numEntries is 60
   Value of numBytes is 2656
24 SQL variables set up. SQLLEN is 21
   and SQLSTR is <select * from empinfo>
   token, representing type of SQL stmt, is 5

```

Figure 52. NDB port client trace with two NDB servers started and one client invoked (Part 5 of 7)

```

29 token, representing an SQL SELECT, is 5
   SQL PREPARE using SQLDA was successful
   SQL DECLARE CURSOR was successful
   SQL OPEN CURSOR was successful
   Storage for one row plus indicator vars obtained
   colBytes is 50. rowBuffer is 58.
   Entering SQLFetch function
   Value of LocalStat is 0
   Value of rowBuffer is 58
   rowBufferp is set at 3bd7fb0
   Value of colBytes is 50
   Top of SQLFetch, RowsInBuff is 0, BufferLeft is 8192
   moveRPDp is 3bd7ff8, ndbrpdln is 0
   Starting Do Forever loop
   rowBufferp storage initialized, moveBufferp is 3bd7fb0, and moveBuffer is 0
   SQL FETCH was successful and have formatted a row
   AnyRows is 1, RowsInBuff is 1, moveBuffer is 50, BufferLeft is 8142
   moveRPDp is 3bd802a, and ndbrpdln is 50
   (rowBuffer is 58, rowBufferp is 3bd7fb0)
   Starting Do Forever loop
   rowBufferp storage initialized, moveBufferp is 3bd7fb0, and moveBuffer is 0
   In SQLFetch, after SQL FETCH, sqlcode is 100...but rows were found
   Entering SQLClose function
   Exiting SQLClose function
   End of query, either by EOQ or by error.
   Before reinitializing:
   rowBufferp is 3bd7fb0, LocalStat is 0, AnyRows is 1
   rowBuffer is 58, colBytes is 50
   Exiting SQLFetch function
   Value of LocalStat is 0
   Value of rowBuffer is 0
   rowBufferp is set at 0
   Value of colBytes is 0
25 Exiting SQLOpen function
   Value of Init_Done is 1
   Value of LocalStat is 0
   Value of rowBuffer is 0
   rowBufferp is set at 0
   Value of colBytes is 0
   Value of numEntries is 60
   Value of numBytes is 2656
   Back from SQLOpen. RC is 0. NDBSRC is 100

```

Figure 52. NDB port client trace with two NDB servers started and one client invoked (Part 6 of 7)


```

26 Exiting program NDBSRV
   Static var NewUser is 0
   NDBC contents at NDBSRV exit is:
   -----
   ndbre1   is 1
   ndbver   is 2
   ndbcb    is NDBC
   ndbsrc   is 100
   ndbappl  is 1
   ndbstat  is 5
   ndbsname is netdbsrv
   ndbusrid is USER1
   ndbpswd  is not echoed in trace
   ndbrqdln is 21
   ndbrqd   is 3bcbf90 (Hex)
   ndbrpdln is 50
   ndbrpd   is 3bd7ff8 (Hex)
27 NDBSRV about to be called on behalf of NDB Client
   18:59:46 EZA4151I MVS NDB SERVER RECEIVED A CALL FROM HOST USERID user1
19 Entering program NDBSRV
   Static var NewUser is 0
   NDBC contents at NDBSRV entry is:
   -----
   ndbre1   is 1
   ndbver   is 2
   ndbcb    is NDBC
   ndbsrc   is 100
   ndbappl  is 1
   ndbstat  is 99
   ndbsname is netdbsrv
   ndbusrid is user1
   ndbpswd  is not echoed in trace
   ndbrqdln is 3
   ndbrqd   is 3bcbfa0 (Hex)
   ndbrpdln is 8192
   ndbrpd   is 3bd7ff8 (Hex)
   ndbrqd contents is:      end
20 NDBC Reply buffer has been initialized
   NDBC Host userid and password verified
   NDBC Control Block header fields verified
30 Processing NDB END command
   Entering DBClose function
   DBUTIL2:DB CLOSE: rtc is 0 and rsc is 0
   Exiting DBClose function
   In NDBSRV:Close: rtc is 0. rsc is 0(Hex).
   CAF CLOSE DB was successful.
   End of END command. NewUser is 1. Return.
16 18:59:13 EZA4150I NDB SERVER STARTED WITH PROGNUM 20000021(HEX), 536870945(DEC)

```

Figure 52. NDB port client trace with two NDB servers started and one client invoked (Part 7 of 7)

Following are short descriptions of the numbered items in the trace.

- 1** This message indicates that the NDB port client is starting up.
- 2** This trace message indicates that tracing is now active.
- 3** The following 10 trace messages show the JCL startup parameters specified at PORTC startup.
- 4** The following two trace messages indicate that the NDB port client was successful at initiating remote procedure call (RPC) communication with the NDB port manager.
- 5** This trace message indicates that the NDB servers run with an RPC timeout value of five minutes.

- 6** The following eight trace messages show the input control block SSCB that will be used when calling the NDB port manager through RPC.
- 7** The following two trace messages indicate that the NDB port client was able to obtain the DB2 subsystem name passed by the parameter DB2SSID= of the PORTC procedure and show what value was obtained.
- 8** This line shows the number of NDB servers specified on the parameter NUMSRV= of the PORTC procedure.
- 9** The following two trace messages indicate that initialization of NDB servers startup has successfully completed.
- 10** The following 10 trace messages indicate that NDB server one was successfully assigned a program number by the NDB port manager. The resulting SSCB contents is also shown.
- 11** The first NDB server has started up. Its assigned program number is shown in hexadecimal and decimal notations.
- 12** The following five trace messages indicate that NDB server one started and shows the values of the parameters passed to it.
- 13** The following 10 trace messages indicate that NDB server two was successfully assigned a program number by the NDB port manager. The resulting SSCB contents is also shown.
- 14** The second NDB server has started up. Its assigned program number is shown in hexadecimal and decimal notations.
- 15** The following five trace messages indicate that NDB server two started and shows the values of the parameters passed to it.
- 16** The NDB server has started up and is waiting to be assigned to an NDB client.
- 17** The following three trace messages indicate that the parameters passed to the NDB server were received and what the values of two of those parameters are. (The third parameter value, program number, is displayed in the previous message.)
- 18** The following two trace messages indicate the NDB server one was assigned to an NDB client and that it has received a request. The host userid the NDB server is to use when sending the user's request to DB2 is user1.
- 19** The following 19 trace messages show the contents of the input NDBC control block as received by NDB server one.
- 20** The following three trace messages indicate that the NDB server is ready to start processing the user's request. The host user ID and password supplied by the user with the NDB client have passed the security check and the NDBC control block was verified as valid.
- 21** The following nine trace messages indicate that this is the first call for this NDB session. The NDB server must establish a connection with DB2. This is accomplished by opening the plan DBUTIL2 using the DB2 Call Attachment Facility (CAF). The open of DBUTIL2 was successful.
- 22** The following nine trace messages show the initial values of various internal control fields used in processing the user's request as they are set at the start of request processing.
- 23** The following five trace messages indicate that this is the first time this

NDB server has been called since being started up. An SQLDA (a control block DB2 uses to pass information back to the NDB server about SQL statements sent to DB2) must be allocated and various initial values set.

- 24** The following three trace messages show the user's SQL statement that will be sent to DB2.
- 25** The following nine trace messages indicate that the user's SQL statement was processed successfully. Also, they show the resulting values of various internal control fields used in processing user requests as they are set at the end of request processing.
- 26** The following 17 trace messages show the contents of the output NDBC control block that is being sent back to the NDB client.
- 27** The following two trace messages indicate that NDB server one has received another request from the NDB client.
- 28** The following five trace messages indicate that NDB server one has been called previously and so only needs to reinitialize certain fields of the SQLDA used by DB2 to pass information to the NDB server.
- 29** The following 33 trace messages show the path taken to process an SQL SELECT statement. The number of messages and their content vary according to the number of rows returned and columns retrieved by the SQL query. This sequence of messages shows that one row at a time is retrieved from DB2, is formatted and is placed in the NDBC reply buffer. The values of various internal fields used to control processing of the SQL query are displayed.
- 30** The following seven trace messages indicate that the user has entered the NDB END command to end this NDB session. NDB server one closes the connection with DB2 by issuing a DB2 CAF close call for the plan DBUTIL2. It was successful. An internal indicator is reset so that the next time this NDB server is invoked, it knows it is starting a new NDB session with a new or different assigned NDB client.

Chapter 19. Diagnosing X Window System and OSF/Motif problems

This chapter describes environment variable XWTRACE that may be useful when diagnosing X Window System and OSF/Motif problems. The environment variable, XWTRACE, controls the generation of traces of the socket level communication between Xlib and the X Window System Server.

- XWTRACE undefined or zero — No trace generated.
- XWTRACE=1 — Error messages.
- XWTRACE>=2 — API function tracing for TRANS functions.

Another environment variable, XWTRACELC, causes a trace of various locale-sensitive routines. If XWTRACELC is defined, a routine flow trace is generated. If XWTRACELC=2, more detailed information is provided.

Note: There are no special post-install activities for GDDMXD in z/OS CS. (GDDM APAR (PN77391) eliminated these activities for TC/IP Version 3 Release 1.) However, if you have an old GDDMXD load library (*tcip.v3r1.SEZALNKG*) in your LNKSTxx member in SYSx.PARMLIB, you need to remove that library from the MVS link list, because it is no longer needed.

Following are some examples of X Window System traces.

Trace output when XWTRACE=2

Figure 53 on page 484 shows a typical stream of socket level activity that is generated when an X application running on z/OS UNIX MVS exchanges information with an X Server.

```

TRANS(OpenCOTSCClient) (/9.2.104.56:0)
TRANS(Open) (1,/9.2.104.56:0)
TRANS(SocketOpenCOTSCClient) (inet,9.2.104.56,0)
TRANS(Connect) (3,/9.2.104.56:0)
TRANS(SocketINETConnect) (3,9.2.104.56,0)
TRANS(GetPeerAddr) (3)
TRANS(ConvertAddress) (2,16,7f9d288)
TRANS(SetOption) (3,2,1)
TRANS(SocketWritev) (3,225bc,1)
TRANS(SetOption) (3,1,1)
TRANS(SocketRead) (3,22344,8)
TRANS(SocketRead) (3,22344,8)
TRANS(SocketRead) (3,7f9d368,224)
TRANS(SocketWrite) (3,7f9eb88,60)
TRANS(SocketRead) (3,2249c,32)
TRANS(SocketRead) (3,2249c,32)
TRANS(SocketRead) (3,2249c,32)
TRANS(SocketWrite) (3,7f9eb88,56)
TRANS(SocketRead) (3,22518,32)
TRANS(SocketRead) (3,22518,32)
TRANS(SocketWrite) (3,7f9eb88,80)
TRANS(SocketWrite) (3,7f9eb88,20)
TRANS(SocketRead) (3,223e8,32)
TRANS(SocketRead) (3,223e8,32)
TRANS(SocketDisconnect) (7f9d2c0,3)
TRANS(Close) (3)
TRANS(SocketINETClose) (7f9d2c0,3)

```

Figure 53. Example of X Application trace output when XWTRACE=2

Each line of the trace provides:

- The name of the function involved from x11trans
- Values of the parameters passed to the function

Trace output when XWTRACELC=2

Figure 54 on page 485 is a partial trace showing typical types of information displayed by locale-sensitive routines.

```

1cPubWrap: XlcCreateLC(C)
1cCT: _XlcAddCT(IS08859-1:GL,"(B)
1cCT: _XlcParseCT
1cCT: _XlcGetCharSetFromEncoding( (B)
1cCT: _XlcParseCT returning: 28 charset 0
1cCharSet: _XlcCreateDefaultCharSet(IS08859-1:GL,""a)
1cCT: _XlcParseCharSet
1cCT: _XlcParseCT
1cCT: _XlcGetCharSetFromEncoding( (B)
1cCT: _XlcParseCT returning: 28 charset 0
1cCharSet: _XlcAddCharSet(IS08859-1:GL)
1cCharSet: _XlcGetCharSet(IS08859-1:GL)
        returned NULL
1cCT: _XlcAddCT    returning: 7f994d8
:
:
(trace statements in this section have been deleted)
1cCT: _XlcAddCT(CNS11643.1986-1:GL,"$(H)
1cCT: _XlcParseCT
1cCT: _XlcGetCharSetFromEncoding( $(H)
1cCT: _XlcParseCT returning: 2428 charset 0
1cCharSet: _XlcCreateDefaultCharSet(CNS11643.1986-1:GL,"""+)
1cCT: _XlcParseCharSet
1cCT: _XlcParseCT
1cCT: _XlcGetCharSetFromEncoding( $(H)
1cCT: _XlcParseCT returning: 2428 charset 0
1cCharSet: _XlcAddCharSet(CNS11643.1986-1:GL)
1cCharSet: _XlcGetCharSet(CNS11643.1986-1:GL)
        returned NULL
1cCT: _XlcAddCT    returning: 7f9c4e0
1cCT: _XlcAddCT(TIS620.2533-1:GR,"-T)
1cCT: _XlcParseCT
1cCT: _XlcParseCT returning: 80 charset 0
1cFile: _XlcResolveLocaleName(C,""," ""}, "2h",)
1cFile: _XlcResolveName(C,/usr/lib/X11/locale/locale.alias)
1cFile: _XlcFileName(7f99420,locale)
1cFile: _XlcResolveLocaleName(C,""," "","")
1cFile: _XlcResolveName(C,/usr/lib/X11/locale/locale.alias)
1cFile: _XlcResolveName(C,/usr/lib/X11/locale/locale.dir)
1cDB: CreateDatabase(/usr/lib/X11/locale/C/XLC_LOCALE)

```

Figure 54. Example of X Application trace output when XWTRACELC=2 (Part 1 of 2)

```

0: XLC_XLOCALE, cs0.ct_encoding,      1: ISO8859-1: GL,
1: XLC_XLOCALE, cs0.wc_encoding,      1: \x00000000,
2: XLC_XLOCALE, cs0.length,    1: 1,
3: XLC_XLOCALE, cs0.side,      1: GL:Default,
4: XLC_XLOCALE, wc_shift_bits,      1: 8,
5: XLC_XLOCALE, wc_encoding_mask,    1: \x00008080,
6: XLC_XLOCALE, state_depend_encoding,      1: False,
7: XLC_XLOCALE, mb_cur_max,    1: 1,
8: XLC_XLOCALE, encoding_name,      1: STRING,
9: XLC_FONTSET, fs0.font,      1: ISO8859-1:GL,
10: XLC_FONTSET, fs0.charset,      1: ISO8859-1:GL,
***
***
    lcdb: _XlcGetResource(7f99420,XLC_XLOCALE,mb_cur_max)
    lcdb: _XlcGetLocaleDataBase(7f99420,XLC_XLOCALE,mb_cur_max)
    lcdb: _XlcGetResource(7f99420,XLC_XLOCALE,state_dependent)
    lcdb: _XlcGetLocaleDataBase(7f99420,XLC_XLOCALE,state_dependent)
    returning NULL
    lcdb: _XlcGetResource(7f99420,XLC_XLOCALE,encoding_name)
    lcdb: _XlcGetLocaleDataBase(7f99420,XLC_XLOCALE,encoding_name)
    returning lcd=7f99420
    lcFile: _XlcResolveLocaleName(C,"",",",",",)
    lcFile: _XlcResolveName(C,/usr/lib/X11/locale/locale.alias)

```

Figure 54. Example of X Application trace output when XWTRACELC=2 (Part 2 of 2)

Each line of trace provides:

- The name of the locale routine.
- The function invoked within that locale routine.
- Where pertinent, charset name or encoding information, or charset name and encoding information..
- If exiting the invoked function, the trace statement indicates that the function is returning.

Chapter 20. Diagnosing Simple Network Management Protocol (SNMP) problems

This section provides explanations for SNMP-related concepts and terms, and includes information on how to diagnose problems.

Overview

The SNMP protocol provides a standardized interface, through which a program on one host (running an SNMP manager) can monitor the resources of another host (running an SNMP agent).

Management information base (MIB)

The information maintained at each agent is defined by a set of variables known as the management information base, or MIB. In addition to the architected list of variables that must be supported by each SNMP agent, an SNMP agent can also support user-defined variables. These user-defined variables that are not part of the architected MIB are known as enterprise-specific MIB variables.

On z/OS CS, the majority of the MIB variables are maintained outside the SNMP agent address space by programs known as SNMP subagents. The subagent program for the TCP/IP-related MIB variables executes in the TCP/IP address space. The subagent program for OMPROUTE-related MIB variables runs as part of OMPROUTE, not as a separate application. The subagent program for SLA-related MIB variables runs as a separate application. For a list of all the MIB objects supported by the agent and subagents shipped as part of z/OS CS, refer to the *z/OS Communications Server: IP User's Guide and Commands*.

In addition, user-written subagent programs can also exist. All subagent programs, whether provided by z/OS CS or user-written, communicate with the SNMP agent over an architected interface known as the Distributed Protocol Interface, or DPI.

When the SNMP agent receives and authenticates a request, it passes the request to the DPI subagent that has registered as the target of the request. You can see this exchange by enabling DPI tracing within the agent.

PDU

The SNMP protocol is based on the exchange of protocol data units, or PDUs, between the SNMP manager and the SNMP agent. SNMP has seven types of PDUs:

GetRequest-PDU

Sent from the manager to request information from the agent.

GetNextRequest-PDU

Requests the next variable in the MIB tree.

GetBulkRequest-PDU

Requests the next variable in the MIB tree and can also be used to specify multiple successors.

GetResponse-PDU

Sent from the agent to return information to the manager.

SetRequest-PDU

Sent from the manager to alter information at the agent.

Trap-PDU

Sent from the agent to report network events to the manager. A trap is an unconfirmed notification.

Inform-PDU

Sent from an agent to a manager or from a manager to another manager to report a network event. Attempts to confirm delivery are made for Inform-PDUs, not Trap-PDUs.

Functional components

The following sections provide detailed descriptions of the SNMP functional components.

Managers

A manager is a client application that requests management data. z/OS CS supports two management applications, the z/OS UNIX SNMP command (**osnmp**) and the NetView SNMP command. The **osnmp** command is a management application used from the z/OS UNIX shell to monitor and control network elements. The NetView SNMP command provides the same type of functions from the NetView environment.

The **osnmp** command runs in a user address space that is created and removed as **osnmp** is issued and completed. The NetView SNMP client requires the following started tasks:

- SNMPIUCV subtask of NetView, which runs in the NetView address space and provides the operator interface to SNMP.
- SNMP query engine address space, which provides the protocol support for the SNMP PDUs.

The SNMPIUCV subtask in the NetView address space and the SNMP query engine address space communicate over an IUCV connection.

Agents

An agent is the server that responds to requests from managers. The agent maintains the MIB. z/OS CS supports a tri-lingual SNMP agent which can understand SNMPv1, SNMPv2C, and SNMPv3 versions of the SNMP protocol. It also communicates with the subagents using DPI1.1 and DPI2.0 protocols.

Subagents

Subagents help the agent by managing a part of the MIB. z/OS CS supports the following subagents:

- TCP/IP subagent that manages TCP/IP-related standard MIB objects and several enterprise-specific MIBs
- OMROUTE subagent that manages the **ospf** MIB
- SLA subagent that manages the **sla** MIB

These subagents communicate with the SNMP agent using the DPI 2.0 protocol.

Trap forwarder daemon

The Trap Forwarder daemon on z/OS CS, listens for SNMP traps on a specified port and forwards them to other configured ports. This eliminates the port contention problem when multiple managers want to receive notifications at the same well-known port (162) at the same IP address.

Definitions

The SNMP agent, subagents and clients must be configured to TCP/IP before use. If the NetView SNMP client is used, Netview configuration is also required.

Several configuration data sets are required. Most of the configuration data can be configured in several places. Details on the syntax of the statements in the files and the search orders for the files are in the *z/OS Communications Server: IP Configuration Reference*. In the text that follows, uppercase file names (such as OSNMP.CONF) indicate the generic name for the file, which can be any of the places in the search order for the file.

TCP/IP configuration files for SNMP are summarized below. For use of the NetView SNMP command, changes are required for the NetView start procedure and the DSIDMN and DSICMD NCCFLST members of the NetView DSIPARM data set. For additional information, refer to the *z/OS Communications Server: IP Configuration Guide*.

osnmp

To use **osnmp**, the following files might be needed:

OSNMP.CONF

Defines configuration data for sending SNMPv1, SNMPv2, and SNMPv3 requests to SNMP agents. You can name this file as either an HFS file or an MVS data set (partitioned or sequential).

MIBS.DATA

Defines textual names for user variables not included in the compiled MIB shipped with the product. You can name this file as either an HFS file or an MVS data set (partitioned or sequential).

SNMP agent

The SNMP agent (osnmppd) uses the following configuration data sets:

OSNMPD.DATA

Defines initial settings for some MIB variables.

PW.SRC

Defines community names, if the SNMPD.CONF file is not being used. Note that community name is a mixed-case, case-sensitive field.

SNMPD.BOOTS

Defines SNMPv3 initialization parameters to the SNMP agent if SNMPv3 security is used.

SNMPD.CONF

Defines security configurations and trap destinations to the SNMP agent. Required if SNMPv3 security is used. May also be used for community-based (SNMPv1 and SNMPv2c) security.

SNMPTRAP.DEST

Defines trap destinations, if the SNMPD.CONF file is not being used.

With z/OS CS, the SNMP agent allows the use of user-based security (SNMPv3) in addition to, or instead of, community-based security (SNMPv1 and SNMPv2c). The choice of configuration data sets depends on the security methods chosen, as shown in Table 32 on page 490.

Table 32. Configuration files and security types

Data set	SNMPv1 and SNMPv2c	SNMPv1, SNMPv2c, and SNMPv3
PW.SRC	Yes	No
SNMPTRAP.DEST	Yes	No
OSNMPD.DATA	Yes	Yes
SNMPD.CONF	No	Yes
SNMPD.BOOTS	No	Yes

TCP/IP subagent

The TCP/IP subagent is controlled by statements in the TCP/IP profile. The following statements are particularly important:

SACONFIG

Defines configuration parameters for the TCP/IP subagent.

ITRACE

Specifies the level of tracing used by the TCP/IP subagent.

OMPROUTE subagent

The SNMP OMPROUTE subagent is controlled by statements in the OMPROUTE configuration file. The following statements are particularly important:

ROUTESA_CONFIG

Defines configuration parameters for the OMPROUTE subagent. You can also use the command **MODIFY ROUTESA**.

OMPROUTE start option -s *n*

Specifies the level of tracing used by the OMPROUTE subagent. You can also use the **MODIFY SADEBUG** command.

OSPF_INTERFACE

Defines an OSPF interface. The OMPROUTE subagent supports only OSPF MIB (RFC 1850).

Note: At least one **OSPF_INTERFACE** must be defined.

SLA subagent

The SLA subagent is controlled by start options specified when the subagent is started. The following statements are particularly important:

PAGTSNMP start option -c *community*

Defines the community name to be used in connecting to the SNMP agent.

PAGTSNMP start option -P *port*

Defines the port to be used in connecting to the SNMP agent.

PAGTSNMP start option -d *n*

Specifies the level of tracing used by the SLA subagent. You can also use the **MODIFY TRACE,LEVEL** command.

SNMP socket call settings

Finally, SNMP makes socket calls that require correct settings in the TCPIP.DATA file. Statements used by SNMP include:

DATASETPREFIX

Can be used in determining the high-level qualifier for agent configuration data sets.

TCPIPJOBNAME

Determines the TCP/IP instance in which SNMP attempts to establish its relationship through the SETIBMOPT socket call. For more information about TCPIPJOBNAME, refer to the *z/OS Communications Server: IP Configuration Reference*.

Trap forwarder daemon

The Trap Forwarder daemon is controlled by the TRAPFWD.CONF file. TRAPFWD.CONF defines the configuration data to forward trap datagrams received on a port to other management applications listening on different ports.

Diagnosing SNMP problems

Problems with SNMP are generally reported under one of the following categories:

- “Abends”
- Connection problems
 - “Problems connecting to the SNMPIUCV subtask” on page 492
 - “Problems connecting the SNMP query engine to the TCP/IP address space” on page 493
 - “Problems connecting the SNMP agent to the TCP/IP address space” on page 493
 - “Problems connecting SNMP agents to multiple TCP/IP stacks” on page 494
 - “Problems connecting subagents to the SNMP agent” on page 495
- Incorrect output
 - “Unknown variable” on page 498
 - “Variable format incorrect” on page 501
 - “Variable value incorrect” on page 502
- “No response from the SNMP agent” on page 503
- “Report received from SNMP agent” on page 504
- “I/O error for SNMP PING” on page 505
- “Traps not forwarded by trap forwarder daemon” on page 505
- “Incorrect address in forwarded trap” on page 506

Note: A nonzero return code from the SNMP agent indicates an abnormal termination. For more information, use the SNMP agent traces sent to the SYSLOGD output.

Use the information provided in the following sections for problem determination and diagnosis of errors reported against SNMP.

For additional information, refer to the *z/OS Communications Server: IP Configuration Guide* and *z/OS Communications Server: IP Configuration Reference*.

Abends

An abend during SNMP processing should result in messages and error-related information being sent to the system console. A dump of the error will be needed unless the symptoms match a known problem.

Documentation

Code a CEEDUMP DD statement in the PROC used to start the SNMP agent to ensure that a useful dump is obtained in the event of an abend.

Analysis

Refer to *z/OS MVS Diagnosis: Procedures* or Chapter 3, “Diagnosing abends, loops, and hangs” on page 21, for information about debugging dumps produced during SNMP processing.

SNMP connection problems

This section describes how to diagnosis and correct SNMP connection problems.

Problems connecting to the SNMPIUCV subtask

Problems in connecting the SNMPIUCV subtask of NetView to the SNMP query engine address space are usually indicated by an error message at the NetView operator console in response to an SNMP request or an attempt to start the SNMPIUCV subtask.

Documentation: The following documentation should be available for initial diagnosis of problems connecting the SNMPIUCV subtask to the SNMP query engine:

- PROFILE.TCPIP data set
- SNMP query engine job output, including SYSPRINT output
- NetView log
- SNMPARMS member of DSIPARMS data set

Analysis: Check for problems connecting the SNMP query engine to the NetView SNMPIUCV subtask:

- Has the SNMP query engine job started successfully?
 - Check the SNMP query engine job output for errors. If the SNMP query engine is started successfully, you should see the message:

```
SQEI001 -- SNMP Query Engine running and awaiting queries...
```

Otherwise, check for errors that might have occurred during socket processing (socket, bind, accept, select, and so on).

- Is the SNMPIUCV subtask started?
 - If not, start the subtask by issuing the command:

```
START TASK=SNMPIUCV
```

from a NetView operator console.

- Was the following message received at the NetView operator console?

```
SNM101W SNMP task (SNMPIUCV) found Query Engine (name) not ready
```

 - Is the *name* that the SNMPIUCV subtask is trying to connect to the correct name for the SNMP query engine address space?
 - If not, check the SNMPARMS member of the DSIPARMS data set to make sure that the value specified for the SNMPQE keyword is the correct SNMP query engine address space name.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation:

- SNMP query engine level-two trace output
- SNMP query engine IUCV communication trace output

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:

- Dump of SNMP query engine address space
- Dump of NetView address space

Information about obtaining a dump can be found in the *z/OS MVS Diagnosis: Tools and Service Aids* manual for your release of z/OS. Obtaining SNMP traces is discussed in “SNMP traces” on page 507.

Problems connecting the SNMP query engine to the TCP/IP address space

Problems connecting the SNMP query engine to the TCP/IP address space are usually indicated by an error message in the SNMP client output, indicating either a socket or IUCV error.

Documentation: The following documentation should be available for initial diagnosis of problems connecting the SNMP query engine to the TCP/IP address space:

- PROFILE.TCPIP data set
- SNMP client output, including SYSPRINT output
- TCPIP.DATA data set

Analysis: Check the following for problems connecting the SNMP client address space to the TCP/IP address space:

- Did any socket-related errors occur?
Check the SNMP query engine job output for socket(), bind(), accept(), or other socket error messages.
- Does job output indicate RC=1011 received for IUCV_CONNECT to *tcip_name*?
Is the *tcip_name* indicated by the IUCV_CONNECT error the correct name for the TCP/IP address space?
 - Is the correct TCPIP DATA data set being used? (The job output should indicate which data set is being used).
 - Is the SYSTCPD DD statement coded in the PROC JCL?

Note: SYSTCPD can be overridden by the global TCPIP.DATA file. Refer to the *z/OS Communications Server: IP Configuration Reference* for additional information about the search order for the TCPIP.DATA data set.

- Does the TCPIPJOBNAME keyword in the TCPIP.DATA data set being used have the correct TCP/IP address space name?

If the problem still occurs after checking the preceding items and making any needed changes, obtain SNMP query engine IUCV communication trace output for problems connecting the client.

The following documentation might also be needed in some cases, but it is suggested that the TCP/IP IBM Software Support Center be contacted before this documentation is obtained:

- Dump of SNMP client address space
- Dump of TCP/IP address space

Information on obtaining a dump can be found in the *z/OS MVS Diagnosis: Tools and Service Aids* manual for your release of z/OS. Obtaining SNMP traces is discussed in “SNMP traces” on page 507.

Problems connecting the SNMP agent to the TCP/IP address space

Problems connecting the SNMP agent to the TCP/IP address space are usually indicated by an error message in the agent traces in the syslog daemon output, indicating a socket error. For more information on reading the syslogd traces, refer to the *z/OS Communications Server: IP Configuration Guide*.

Documentation: The following documentation should be available for initial diagnosis of problems connecting the SNMP agent to the TCP/IP address space:

- PROFILE.TCPIP information
- SNMP agent tracing (at level 255) to the syslog daemon output
- TCPIP.DATA information
- OMVS console output for any command responses and traces

Analysis: Check the following for problems connecting the SNMP client or agent address space to the TCP/IP address space:

- Are you connected to the correct TCP/IP address space? This is obviously a concern when running multiple stacks. See “Problems connecting SNMP agents to multiple TCP/IP stacks”.

If you get a message “unable to connect to TCPIP JOBNAME,” you are not connected to the correct address space. If you have defined two or more stacks, make sure your TCPIPjobname in the TCPIP.DATA data set used by the SNMP agent matches the NAME field on the SUBFILESYSTYPE statement for ENTRYPOINT(EZBPFIN) in the BPZPRMxx member you used to start z/OS UNIX MVS.

- Did any socket-related errors occur?

Check the SNMP agent syslogd for socket(), bind(), accept(), or other socket error messages. For example, a bind() failure will occur when one or more of the ports needed by the SNMP agent is already in use. Refer to the *z/OS Communications Server: IP Configuration Guide* for more information about syslogd.

- Is the correct TCPIP.DATA information being used? Is the SYSTCPD DD statement coded in the PROC JCL? Is the RESOLVER_CONFIG environment variable passed on the SNMP agent initialization parameters?

Refer to the *z/OS Communications Server: IP Configuration Reference* for additional information about the search order for the TCPIP.DATA data set.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation for problems connecting the agent.

- Dump of SNMP agent address space.
- Dump of OMPROUTE address space (for OMPROUTE subagent problems).
- Dump of SLA subagent address space (for SLA subagent problems).
- Dump of TCP/IP address space.
- The syslogd traces from the agent (using trace level 255). Refer to the *z/OS Communications Server: IP Configuration Guide* for more information about reading the syslogd.

Information on obtaining a dump can be found in the *z/OS MVS Diagnosis: Tools and Service Aids* manual for your release of MVS. Obtaining SNMP traces is discussed “SNMP traces” on page 507.

Problems connecting SNMP agents to multiple TCP/IP stacks

To receive TCP/IP related management data, each TCP/IP stack that is started must run its own SNMP agent. This requires that each agent can find the TCP/IP job name of the TCP/IP stack that it wants to associate with.

Analysis: Check the following for problems connecting the SNMP agent to the correct TCP/IP stack:

- Message EZZ6205I indicates that when _iptcpn() was called, it did not return the correct TCPIPjobname for that agent.
 - Check _iptcpn()'s search path.
 - Check to see if the _BPXK_SETIBMOPT_TRANSPORT environment variable has been set in the cataloged procedure.

Refer to the *z/OS Communications Server: IP Configuration Reference* for additional information..

- Message EZZ6272I indicates that the setibmopt call failed. This means that _iptcpn() returned a name that z/OS UNIX did not recognize as a PFS. Check the BPXPRMxx member (in SYS1.PARMLIB) used to configure z/OS UNIX.

Problems connecting subagents to the SNMP agent

Problems connecting an SNMP subagent to the SNMP agent are generally indicated by one of the following:

- A socket error at the subagent.
- Authentication failures when the subagent attempts to open a connection.
- A “no such name” response from the SNMP agent when an SNMPv1 manager requests a variable owned by the subagent.
- A “no such object” response from the SNMP agent when an SNMPv2 or SNMPv3 manager requests a variable owned by the subagent.

Documentation: The following documentation should be available for initial diagnosis of interface connection problems:

- PROFILE.TCPIP information.
- SNMP agent job output, including syslogd output.
- TCP/IP subagent syslogd output obtained by specifying the profile statement ITRACE ON SUBAGENT 2 (if the subagent is the TCP/IP subagent).
- Output of the Netstat HOME/-h command.
- TCPIP.DATA information.
- OMPROUTE subagent syslogd output obtained by starting OMPROUTE with the -s1 option or by issuing the MODIFY SADEBUG command to start OMPROUTE subagent tracing (if the subagent is the OMPROUTE subagent).
- SLA subagent syslogd output obtained by starting the SLA subagent with the -d 2 option or by issuing the MODIFY TRACE,LEVEL command to start SLA subagent tracing (if the subagent is the SLA subagent).

Analysis: Check the following for problems connecting an SNMP subagent program to the SNMP agent:

1. Is the subagent in question the TCP/IP subagent? If so,
 - Is the SACONFIG statement configured correctly?
 - Is SACONFIG disabled?
2. Is the subagent in question the OMPROUTE subagent?
 - Is the OMPROUTE ROUTESA_CONFIG statement configured correctly?
 - Is the OMPROUTE subagent (ROUTESA) disabled?
 - Does the port number match the SNMP agent and OMPROUTE application for the OMPROUTE ROUTESA_CONFIG parameter AGENT=<agent port number>?
 - Does the community name (or password) match with the SNMP agent and OMPROUTE application for the OMPROUTE ROUTESA_CONFIG parameter COMMUNITY=<community string>?
3. Is the subagent in question the SLA subagent?

- Does the port number specified on the -P parameter of the SLA subagent match the port number specified by the SNMP agent?
 - Does the community name (or password) specified on the -c parameter of the SLA subagent match the community name (or password) specified by the SNMP agent?
4. If you are using an *h/q*.HOSTS.SITEINFO file (or its HFS equivalent, */etc/hosts*), you must ensure that the IP address in this file for the system on which the agent/subagent are executing matches an interface IP address of the TCP/IP stack to which the agent/subagent are connected. The interface IP addresses for a TCP/IP stack are defined on the HOME profile statement.
 5. Is the subagent using the correct IP address to send the connection request to the SNMP agent? The subagent uses the primary interface IP address of this stack when sending the connection request to the SNMP agent. The primary interface IP address is either the first IP address in the HOME list or the IP address specified on a PRIMARYINTERFACE TCP/IP profile statement. Check the Netstat HOME/-h output to verify the primary interface address of the stack. This IP address is the one that will be used by the SNMP agent, along with the community name to verify the subagents authority to connect to the SNMP agent.
 6. Is the port number correct?
 7. Is the community name (or password) correct?

Note: Note that community name is a mixed-case, case-sensitive field. Many times the client cannot get a response from an agent because the agent has a community string of PUBLIC. Most clients default their community string to *public*.

8. If the SNMP agent is configured for SNMPv3, is the community name configured in the agent SNMPD.CONF file? The subagent can use the community name only if VACM_GROUP, VACM_VIEW, and VACM_ACCESS are defined. For the subagent to connect, the VACM_VIEW must include the dpiPort objects.
9. Did any socket-related errors occur?
Check the SNMP agent/subagent syslogd for socket(), bind(), accept(), or other socket error messages, particularly error messages related to the DPI connection.
10. If the subagent is using TCP to connect to the SNMP agent then the connection could have been closed by the agent due to a security authorization failure. If the agent security resource name has been defined in the SERVAUTH class, then the subagent must be running on the same TCP/IP stack as the agent and the user ID of the subagent must be permitted to the resource name in order for the connection to succeed. See the SNMP chapter of the *z/OS Communications Server: IP Configuration Guide* for a description of the agent security resource name used with TCP connections between SNMP agent and subagent.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation:

- SNMP agent 255 (trace all) output.
- If the problem is with the TCP/IP subagent, get the subagent traces. These are turned on by specifying the ITRACE statement in the PROFILE.TCPIP file. This can be done as part of the initial TCP/IP startup. It can also be done after TCP/IP has been started by using the VARY TCPIP command, which is documented in the *z/OS Communications Server: IP Configuration Reference*.

- If the problem is with the OMPROUTE subagent, get the OMPROUTE subagent traces. Turn these on by starting OMPROUTE with the -s1 option or by issuing the MODIFY SADEBUG command to start OMPROUTE subagent tracing.
- If the problem is with the SLA subagent, get the SLA subagent traces. Turn these on by starting the SLA subagent with the -d 2 option or by issuing the MODIFY TRACE,LEVEL command to start SLA subagent tracing.
- If the problem is with a user-written subagent program, use the DPIdebug() DPI library routine to collect dpi traces in the user-written subagent program. DPIdebug() sends output to the syslogd.

The following is a list of things to look for in the SNMP agent trace:

- One of the following incoming SNMP GetRequest-PDU:
 - dpiPortForTcp (1.3.6.1.4.1.2.2.1.1.1) for TCP connect. This is caused by DPIconnect_to_agent_TCP
 - dpiPathNameForUnixStream (1.3.6.1.4.1.2.2.1.1.3) for UNIX connect. This is caused by DPIconnect_to_agent_UNIXstream
- Some questions to consider:

- Was the GetRequest-PDU received? If the GetRequest was not received, was it sent to the right port?

In the case of the TCP/IP subagent, the value of the AGENT keyword on the SACONFIG statement in the profile must match the value of -p that was specified (or defaulted) when the agent was invoked.

- Does it have a valid community name in the request?
 - SNMP subagents must use a valid (including correct case) community name as defined in the PW.SRC data set (or *hlq.SNMPD.CONF* data set when using SNMPv3 security) when requesting the dpiPort or dpiPath variable.
 - Note that community name is a mixed-case, case-sensitive field. Specify as follows:
 - For the TCP/IP subagent, specify the community name in the SACONFIG statement.
 - For the OMPROUTE subagent, specify the community name in the ROUTESA_CONFIG statement.
 - For the SLA subagent, specify the community name by way of the -c parameter.
- If SNMPv3 is being used, the community name must be defined in the VACM_GROUP statement in the SNMPD.CONF file for the SNMP agent. A VACM_ACCESS statement also needs to be defined to give that group read access to a VACM_VIEW that includes dpiPort objects.
- dpiPathNameForUnixStream defaults to /tmp/dpi_socket and provides an HFS pathname used in connecting a DPI subagent with the SNMP agent. The default can be overridden by using the -s parameter when starting the agent or by adding an entry for dpiPathNameForUnixStream in the OSNMPD.DATA file.

A user-written subagent running from a nonprivileged user ID needs write access to the file. Otherwise, a subagent using DPIconnect_to_agent_UNIXstream() would have to be run from an OMVS superuser user ID or other user ID with the appropriate privilege.

- Outgoing GetResponse-PDU for the dpiPort variable:
 - Was the SNMP GetResponse-PDU sent back to the SNMP subagent?
 - Was it sent to the correct IP address?
 - Did it have the correct value for the DPI port?

- The actual value for the DPI port for TCP can be determined by issuing an `onetstat -A` command at the SNMP agent. This will display the port on which the agent is accepting incoming UDP requests.
 - To display the `dpiPath` name, issue an `osnmp get` request for `dpiPathNameForUnixStream`.
 - One of the following incoming subagent connections:
 - Message `EZZ6244I Accepted new DPI inet subagent connection on fd fd=xx from inet address xxxx port xxxx.`
 - `EZZ6246I Accepted new DPI inet socket connection on fd=xx`
- Note:** `fd=xx` is the number associated with this specific subagent connection. Use it to correlate with later DPI trace messages. The name and number of the port `xxxxx port xxxx`.
- DPI packets transferred for this FD number

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:

- Dump of SNMP agent address space
- Dump of TCP/IP address space (for TCP/IP subagent problems)
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of SLA subagent address space (for SLA subagent problems)
- Dump of user SNMP subagent address space
- Trace from subagent in `syslogd`

Information on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP agent traces is discussed in “Starting SNMP agent traces” on page 508.

Incorrect output

Unknown variable

Unknown variable problems are indicated by a **noSuchName** or **noSuchObject** response on an SNMP request. The **noSuchName** response indicates an error returned on an SNMPv1 request. For SNMPv2 and SNMPv3, more specific errors are returned, such as **noSuchObject** and **noSuchInstance**.

Unknown variable problems are usually caused by one of the following:

- A typographical error in the name or OID
- An incorrect instance number

Note: A common mistake is forgetting to add a dot-zero (.0) to an object that is not part of a table. For example, a GET for `ifNumber` returns with “no such object” while performing a GET on `ifNumber 0`. If unsure how to proceed, try a GETNEXT on `ifNumber`. This will return the dot-zero (.0) version of the OID along with its value.

- The subagent supporting the MIB object is not started or is not completely connected to the SNMP agent.
- When SNMPv3 is configured, the object is not within the MIB view the user or community can access.

When the NetView SNMP client is used, unknown variable problems are reported when the SNMP client receives either a major error code 2 (internally detected

error), minor error code 7 (unknown MIB variable), or a major error code 1 (SNMP agent reported error), minor error code 2 (no such name) in response to an SNMP request.

Documentation: The following documentation should be available for initial diagnosis of unknown variable problems:

- SNMP syslogd output with traces for both the agent and subagent. Refer to the *z/OS Communications Server: IP Configuration Guide* for more information about syslogd.
- MIBS.DATA, when osnmp is used.
- SNMP query engine job output, when NetView SNMP is used.
- NetView log, when NetView SNMP is used.
- *hlq.MIBDESC.DATA* data set, when NetView SNMP is used.
- If SNMPv3 security is being used, the SNMP agent configuration file (SNMPD.CONF). If the osnmp command is the client being used, the osnmp command configuration file (OSNMP.CONF) may also be needed.
- Include all the configuration files described earlier under “Definitions” on page 489.

Analysis: Check the following for unknown variables at the SNMP agent:

1. Was the variable requested with the correct instance number?
Variables that are not in a table have an instance number of 0. Variables that are part of a table may have more than one occurrence of the variable value. To get the value of the variable, you will need to request a specific instance of the variable. To find the instance number, issue a GET NEXT request; the first occurrence of the variable should be returned.
2. If the variable is not defined in any compiled MIB, is the variable name included in the MIBS.DATA file for the osnmp command) or the *hlq.MIBDESC.DATA* file (for the Netview SNMP command)?
3. Did the DPI connection come up successfully?
 - a. Check the SNMP agent job output for messages indicating a problem in create_DPI_port.
 - b. If the DPI port was not successful, no SNMP subagents will be able to register MIB variables. The SNMP agent will have no knowledge of these unregistered variables and will report them as “noSuchName” for SNMPv1 requests or “noSuchObject” for SNMPv2 and SNMPv3 requests.
4. Has the subagent successfully connected to the SNMP agent?
 - a. For subagents shipped as part of z/OS CS, check the MVS operator console for a message indicating that the subagent has completed its initialization.
 - b. Issue an osnmp walk command on the SNMP agent subagent status table. For example, either of the following commands display the subagents that are connected to the z/OS CS SNMP agent and the status of their connections:
 - osnmp -v walk saDescription
 - osnmp -v walk saStatus

A value of 1 for saStatus indicates that the subagent connection to the SNMP agent is valid. Following are other possible status values:

invalid	(2)
connecting	(3)
disconnecting	(4)
closedByManager	(5)
closedByAgent	(6)
closedBySubagent	(7)
closedBySubAgentTimeout	(8)
closedBySubAgentError	(9)

5. If the SNMP agent was configured with SNMPv3 security, is the object within the MIB view of that allowed for that user or community?
 - a. Look at the agent SNMPD.CONF file to determine to which VACM_GROUP the user or community name on the failing request belongs. Then examine the VACM_ACCESS statements for that group for the level of security requested on the failing request to determine which MIB views have been permitted to the user or community name.
 - b. Alternatively, SNMP agent configuration can be determined from SNMP agent traces if they were set to level 255 at agent initialization.
 - c. SNMP agent configuration can also be determined dynamically by issuing osnmp walk requests against the agent configuration MIB objects, such as the vacmSecurityToGroupTable and the vacmAccessTable. Reading the values in these tables requires an understanding of how the tables are indexed. Refer to Requests for Comments (RFCs) 2573, 2574, and 2575 for an explanation of the MIB objects containing the SNMP agent configuration.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation:

For variable not recognized by manager messages:

- If the manager is the osnmp command, use the -d 4 flag to get level four manager traces.
- If the manager is the NetView SNMP command, obtain the SNMP query engine level two output.

The SNMP query engine level two trace shows the information flowing between the SNMP query engine and the SNMPIUCV subtask of NetView. Verify in the trace that the variable name being requested is being passed correctly to the SNMP query engine.

For agent unknown variable:

- SNMP agent level 15 trace output
- Traces from SNMP subagent programs (if the variable is supported by a z/OS CS subagent)

The SNMP agent level 15 trace will show PDUs between the manager and agent, as well as between the agent and any existing subagents. Things to look for in the trace:

1. Is the ASN.1 number received from the manager in the SNMP GetRequest-PDU correct?
2. Has a DPI packet registering the requested variable been received?
 - a. If not, if you know which subagent program owns the variable, check the subagent program for errors.
 - b. If the DPI register has been received, make a note of the FD number for further trace information.
3. Were any errors reported for this FD number after the DPI register request was received?
4. Was there a DPI information exchange over this FD number as a result of the incoming SNMP GetRequest-PDU?

Another approach to this problem is to look at the agent saMIB variables. This information can be useful when traces are not available. The saMIB variables include the following information:

- An entry for each subagent (including a field for subagent status)

- A table of all trees registered, including:
 - Subagent to which the tree is registered
 - Status of the tree (valid, not valid, and so on)

A description of the saMIB objects can be found in the file `samib.mib` in the `/usr/lpp/tcpip/samples` directory.

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:

- Dump of SNMP agent address space
- Dump of TCP/IP address space (for z/OS CS subagent variables)
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of SLA subagent address space (for SLA subagent problems)
- Dump of user subagent address space

Information on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP traces is discussed later in “SNMP traces” on page 507.

Variable format incorrect

Problems with incorrectly formatted variables are generally reported when the variable value from the GetResponse-PDU is displayed at the manager in the incorrect format (for example, as hexadecimal digits instead of a decimal value or a display string).

Documentation: The following documentation should be available for initial diagnosis of incorrectly formatted variables:

- MIBS.DATA, when `osnmp` is used
- NetView log, when the NetView SNMP command is used
- `hlq.MIBDESC.DATA` data set, when NetView SNMP is used

Analysis: Check the following:

1. Is the variable contained in the `hlq.MIBDESC.DATA` data set or `MIBS.DATA` file?
 - a. The SNMP query engine will use the `hlq.MIBDESC.DATA` data set to determine the display syntax of the variable value. NetView SNMP requires that all MIB object names be included in the `hlq.MIBDESC.DATA` data set.
 - b. `osnmp` searches the `MIBS.DATA` file for a MIB name definition. If it is not found, the value in the compiled MIB is used.
2. Is the value listed in the syntax position of the `hlq.MIBDESC.DATA` data set or `MIBS.DATA` file record for this variable the correct syntax?

Note that the value specified for syntax (for NetView) is case-sensitive and must be specified in lowercase.

3. For NetView SNMP, is the variable value type specified in message `SNM043I` Variable value type: correct?

Refer to the *z/OS Communications Server: IP User's Guide and Commands* section about “Managing TCP/IP Network Resources Using SNMP” for the meanings of the variable value types.

If the problem still occurs after checking the preceding and making any needed changes, obtain the following documentation:

- For the TCP/IP subagent, subagent ITRACE level four output to show that the subagent returned to the SNMP agent.

- For the OMPROUTE subagent, syslogd output obtained by starting OMPROUTE with the -s1 option or by issuing the MODIFY SADEBUG command to start OMPROUTE subagent tracing.
- For the SLA subagent, syslogd output obtained by the SLA subagent with the -d 2 option or the MODIFY TRACE,LEVEL command to start SLA subagent tracing.
- For user-written subagents, DPIDebug(2) output, which is sent to the syslogd. For more information on reading the syslogd traces, refer to the *z/OS Communications Server: IP Configuration Guide*.
- SNMP query engine level four trace output or osnmp command trace level four.
- SNMP manager command output showing incorrectly formatted variable.
- SNMP agent level 31 trace output shows the DPI packet exchanges between the agent and subagent, as well as the value returned to the manager.

In the traces, verify that the variable value and syntax are passed correctly in the SNMP GetResponse-PDU from the agent to the SNMP manager.

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:

- Dump of SNMP subagent address space
- Dump of SNMP agent address space
- Dump of SNMP query engine address space
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of SLA subagent address space (for SLA subagent problems)

Information on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP traces is discussed in “SNMP traces” on page 507.

Variable value incorrect

Problems with incorrect variable values are generally reported when the variable value from the GetResponse-PDU displayed at the manager contains incorrect information.

Documentation: The following documentation should be available for initial diagnosis of variables with incorrect values:

- SNMP agent syslogd trace output.
- If the object is supported by the TCP/IP subagent, the syslogd output. Obtain the syslogd output using the profile statement ITRACE ON SUBAGENT 4.
- MIBS.DATA, if osnmp is being used.
- *hlq*.MIBDESC.DATA, if NetView SNMP is being used.
- NetView log, if NetView SNMP is used.

Analysis: Check the following:

1. Is the object identifier in the MIB description file correct?
2. Were any errors reported at the SNMP agent when the variable was requested?
3. Is the variable being cached at the SNMP query engine?

The SNMP query engine uses the *hlq*.MIBDESC.DATA data set to determine the length of time to cache the variable value (or a default time length if the variable is not found in the *hlq*.MIBDESC.DATA data set). If the variable is requested before the caching time is up, the cached value is used instead of obtaining a new value.

4. Is the variable cached at the TCP/IP subagent?

The TCP/IP subagent caches variable values for the length of time specified by the *ibmMvsSubagentCacheTime* MIB object, set by default to 30 seconds.

5. Is the variable supported by the SLA subagent? If so, is it being cached? The SLA subagent caches MIB objects for five minutes by default, but the cache time can be overridden at subagent initialization time with the `-t` parameter.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation:

- SNMP agent level three showing what was returned to the client.
- For the TCP/IP subagent, ITRACE level four trace output showing what the subagent returned to the SNMP agent.
- For the OMPROUTE subagent, syslogd output obtained by starting OMPROUTE with the `-s1` option or by issuing the MODIFY SADEBUG command to start OMPROUTE subagent tracing
- For the SLA subagent, syslogd output obtained by the SLA subagent with the `-d 2` option or the MODIFY TRACE,LEVEL command to start SLA subagent tracing.
- For user-written subagents, DPldebug(2) output which is sent to the syslogd. For more information on reading the syslogd traces, refer to the *z/OS Communications Server: IP Configuration Guide*.

In the traces, verify that the variable value is passed correctly from the SNMP subagent to the SNMP agent and from the SNMP agent to the client.

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:

- Dump of TCP/IP address space (for z/OS CS subagent variables).
- Incorrect values from the TCP/IP subagent are probably due to an error in the TCP/IP stack. In this case, a dump of the TCP/IP address space and a CTRACE from the engine might be useful. You can also use the `onetstat` command to verify that the TCP/IP subagent is reporting what the TCP/IP engine believes the value to be.

Information on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP traces is discussed in “SNMP traces” on page 507.

No response from the SNMP agent

Problems receiving a response from the SNMP agent are generally reported when an SNMP request is issued from a manager but no response from the agent is received. This is usually reported as a timeout message.

Documentation

The following documentation should be available for initial diagnosis when no response is received from the agent:

- OMVS console output for any command responses and traces, if `osnmp` is being used.
- NetView console output or command responses if NetView SNMP is used.
- SNMP agent syslogd output.
- The OSNMP.CONF file (if the `osnmp` command is the manager).
- PW.SRC or SNMPD.CONF file being used by the SNMP agent.

Analysis

Check the following when no response is received from an agent:

1. Is the SNMP agent running?
2. Is a path to the agent available? Try issuing a PING request to the IP address of the agent.

3. What is the timeout value? For example, the timeout value on the `osnmp` command defaults to three seconds. Trying the request again with a larger timeout value, such as 15 seconds, may result in an answer.
4. Does the request use the correct port number and IP address?
5. Were any errors reported at the SNMP agent when the variable was requested?
6. If community-based security is being used, is the correct community name (including correct case) being used in the request?
7. If the manager is the `osnmp` command, was the command destination specified using the `-h` parameter? Did the `-h` value have an entry in the `OSNMP.CONF` file? If so, and the destination agent only supports SNMPv1, the request will be discarded. Try reissuing the command using an IP address as the value of the `-h` parameter. If an entry with the value specified on the `-h` parameter does not exist in the `OSNMP.CONF` file, an SNMPv1 request will be sent. An SNMPv2 or SNMPv3 agent can process SNMPv1 requests, but an SNMPv1 agent cannot process SNMPv2 or SNMPv3 requests.

If the problem still occurs after checking the preceding items and making any needed changes, obtain SNMP agent level seven trace output documentation.

Check the following in the SNMP agent traces:

1. Was the SNMP request PDU received by the agent?
2. Did it have a valid community name? Note that community name is case-sensitive and mixed-case.
3. Was the IP address of the manager the expected IP address?
4. Was an SNMP GetResponse-PDU sent back to the manager?
5. Was an AuthenticationFailure trap generated?

Note: For these traps to be generated, you must first provide the trap destination information in either the `SNMPTRAP.DEST` or `SNMPD.CONF` file, then set the `snmpEnableAuthenTraps` MIB variable to 1, signifying traps are enabled. For detailed information on enabling traps, refer to the *z/OS Communications Server: IP Configuration Reference*.

The following documentation might also be needed in some cases, but contact the IBM Software Support Center before this documentation is obtained:

- Dump of SNMP agent address space
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of SLA subagent address space (for SLA subagent problems)

Information on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP traces is discussed in “SNMP traces” on page 507.

Report received from SNMP agent

With SNMPv3, certain error conditions detected on a request are sent back from the SNMP agent to the SNMP manager as a report. Some reports are expected as part of normal processing, but most often they indicate an error condition.

For the `osnmp` command, some reports occur during normal processing, such as the `usmStatsUnknownEngineIDs` condition, which occurs as the `osnmp` command performs discovery processing to learn the SNMP engineID of the agent with which it is communicating. Normal processing reports are not displayed by `osnmp` unless debug tracing is active. Reports that indicate error conditions are typically displayed using the EZZ33431 message. For example, when an attempt is made to use a USM user with an authentication key that does not match the key that is configured at the SNMP agent, the `usmStatsWrongDigests` report will be received.

Figure 57 on page 511 shows the output received by an SNMP manager when the authentication key sent by an `osnmp` command did not match the key defined at the agent. The command issued in the z/OS UNIX shell was:

```
$ osnmp -h v374 -v walk usmUserStatus
```

```
EZZ33431 Report received : usmStatsWrongDigests  
EZZ33011 Error return from SnmpRecvMsg()
```

Following are other common reports:

usmStatsUnknownUserNames

Indicates a request was received for a user that is not defined at the SNMP agent.

usmStatsUnsupportedSecLevels

Indicates a request was received for a defined user, but the user was not configured at the SNMP agent to use the security level specified in the request.

usmStatsDecryptionErrors

Indicates an encrypted request was received at the SNMP agent, but the request could not be decrypted. This can be the result of an invalid privacy key.

I/O error for SNMP PING

NetView users can issue a PING request using SNMP PING. SNMP I/O error problems are reported when a major return code of 2 (internally-detected error) and a minor return code of 4 (some I/O error occurred) are received when issuing an SNMP PING. This type of problem is generally caused by an error in the PROFILE.TCPIP data set.

Documentation

The PROFILE.TCPIP data set should be available for initial diagnosis of SNMP I/O problems.

Additional documentation that might be needed later is discussed in “Analysis”.

Analysis

Obtain the following documentation:

- SNMP query engine job SYSPRINT output
- SNMP query engine level two trace output
- SNMP query engine IUCV communication trace output

The following documentation might also be needed in some cases, but it is suggested that TCP/IP customer support be contacted before this documentation is obtained:

- Dump of SNMP address space
- TCP/IP packet trace

Information on obtaining a dump can be found in the *z/OS MVS Diagnosis: Tools and Service Aids* manual for your release of MVS. Obtaining SNMP traces is discussed “SNMP traces” on page 507.

Traps not forwarded by trap forwarder daemon

Problems with traps not getting forwarded by the trap forwarder daemon are most likely the result of configuration errors or problems in the network.

Documentation

The following documentation should be available for initial diagnosis:

- TRAPFWD.CONF file
- Trapfwd traces, level three
- Traces from the sending agent (the originator of the trap)
- Trace from the receiving client (the target of the forwarded trap)

Analysis

Check the following:

1. Is the target address correctly configured in the TRAPFWD.CONF file?
If the target is designated by a host name, check the trapfwd trace to determine whether or not the hostname was correctly resolved to an IP address.
2. Is the trap being received at the trap forwarder daemon?
If trapfwd traces indicate the trap is not being received at the trapfwd daemon, examine traces from the SNMP agent sending the trap. Determine whether or not the SNMP agent did in fact send the trap.
3. Are there network problems between the trap forward daemon and the target client?
By issuing an SNMP GET request at the target client to the SNMP agent on the same host as the trap forward daemon, you can determine whether or not UDP packets are correctly reaching the client.
4. Are the UDP packets being discarded due to congestion at the TCP/IP stack?
If the trapfwd trace indicates that the trap is correctly being sent from the trap forwarder daemon to the target client, but the trap is not being received, consider setting NOUDPQueueLimit on the UDPCONFIG statement. This is used to specify that UDP should not have a queue limit and would prevent traps from being lost due to congestion.

If the above analysis does not correct the problem, the following documentation should be gathered and the IBM Software Support Center should be contacted:

- UDP packet trace on the TCP/IP stacks where the originating SNMP agent, the trap forwarder daemon, and the target client are running.

Incorrect address in forwarded trap

Documentation

The following documentation should be available for initial diagnosis:

- TRAPFWD.CONF file
- Trapfwd traces, level 3
- Traces from the sending agent (the originator of the trap)
- Trace from the receiving client (the target of the forwarded trap)

Analysis

Check the following:

1. What is the version of the SNMP trap?
In the case of SNMPv1 traps, the address from which the trap originated is encoded within the trap packet. A manager that needs the originating address should look into the SNMP packet to get the address. In the case of SNMPv2 traps, the originating address is not encoded within the trap PDU. If a manager uses the address from which the trap packet is received, it would not be the originating address but the address at which the trap forwarder daemon is running. If the manager needs the originating address in the case of SNMPv2

traps, the trap forwarder should be configured to append the originating address to the trap and the manager should be capable of reading the address from the end of the received trap packet. For more information on the format in which the address is appended, refer to the *z/OS Communications Server: IP User's Guide and Commands*.

2. Is it a SNMPv2 trap?

Check to see if the ADD_RECVFROM_INFO is specified correctly in the TRAPFWD.CONF file. If it is not specified, then add the option to the configuration file. Note, the receiving manager must be capable of processing the RECVFROM information at the end of the trap packet.

If the above analysis does not correct the problem, collect the above documentation and contact the IBM Software Support Center.

SNMP traces

There are several types of traces that can be useful in identifying the cause of SNMP problems:

- Manager traces
- Agent traces
- Subagent traces
- TRAPFWD traces

These traces are discussed in the following sections.

Starting manager traces

To obtain traces when the SNMP manager being used is the osnmp command, issue osnmp with the -d option. You can specify a trace level of zero to four. A trace level of zero provides no tracing, while a level four provides the most. Tracing for osnmp is done on a per-request basis. Traces return to the console, but they can be redirected to a file issuing the OMVS redirect operand (>).

When NetView SNMP is being used, traces for the SNMP Query Engine can be obtained by starting the SNMP Query Engine and specifying -d trace_level where trace_level is one of the following:

- 1 Display errors.
- 2 In addition to 1, also display SNMP query engine protocol packets exchanged between the SNMP query engine and the SNMPIUCV subtask, with the exception of TRAP packets sent to NetView from the query engine.
- 3 In addition to 2, also display decoded SNMP protocol packets sent and received along with some additional informational messages.
- 4 In addition to three, display the BER-encoded packets received from NetView or from an SNMP agent. Also, add display of SNMP query engine protocol packets for TRAPs sent from the query engine to NetView.

For example:

```
//SNMPQE EXEC PGM=SQESERV,REGION=4096K,TIME=1440,PARM='-d 3'
```

Also, the -it option can be used to obtain a trace of IUCV communication.

SNMP Query Engine trace output is sent to the SYSPRINT DD specified in the Query Engine JCL.

Starting SNMP agent traces

If agent is not running

If the SNMP agent is not already running, specify the -d parameter when you invoke the agent. You can start the SNMP agent in one of two ways:

- Through the start options in the JCL used to start the SNMP agent (more common). For example,

```
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=1440,PARM='-d 8'
```

- Through OMVS, using the osnmpd command. For example:

```
osnmpd -d 255 &
```

Use one of the following trace levels or a combination of them:

- | | |
|-----|--------------------------------|
| 1 | Trace SNMP requests |
| 2 | Trace SNMP responses |
| 4 | Trace SNMP traps |
| 8 | Trace DPI packets level 1 |
| 16 | Trace DPI internals level 2 |
| 32 | Internal trace (debug level 1) |
| 64 | Extended trace (debug level 2) |
| 128 | Trace DPI internals level 2 |

Combining trace levels: To combine trace levels, add trace level numbers. For example, to request SNMP requests (level 1) and SNMP responses (level 2), you would request trace level 3.

Trace records are sent to the file specified by the daemon.debug entry in the SYSLOG configuration file. For more information refer to the *z/OS Communications Server: IP Configuration Guide*.

If agent is already running

You can use the MVS MODIFY command to start and stop trace dynamically. Use of this support is restricted to the users with MODIFY command privilege.

If you start the agent from JCL, you have no difficulty knowing the procname. However, if you start the agent from OMVS, the agent generates a message to syslogd. This message indicates the job name the agent is running; this is the job name to specify on the MODIFY command.

Note: While most agent tracing can be modified dynamically, user-friendly formatting of SNMPv3 requests in and responses out can be enabled only at agent initialization.

For example, assume the procname is OSNMPD and you want to change the trace level to 3 (tracing SNMP requests and responses). You would enter

```
MODIFY OSNMPD,trace,level=3
```

For more information on using the MVS MODIFY command, refer to the *z/OS Communications Server: IP Configuration Reference*.

Starting TCP/IP subagent traces

To start the TCP/IP subagent traces, code the ITRACE statement in the PROFILE.TCPIP file or in a file to be used by the *z/OS Communications Server: IP Configuration Reference*.

```
ITRACE ON SUBAGENT level
```

where *level* is one of the following values:

- 1 General subagent trace information.
- 2 General subagent trace information plus DPI traces.
- 3 General subagent trace information plus extended dump trace. This level provides storage dumps of useful information, such as storage returned by the IOCTL calls.
- 4 General subagent trace information, plus extended dump trace and DPI traces.

The trace output is sent to the syslogd. Trace records are found in the file specified by the daemon.debug entry in the SYSLOG configuration file. For more information refer to the *z/OS Communications Server: IP Configuration Guide*.

To stop TCP/IP subagent traces, code the ITRACE statement in the PROFILE.TCPIP or the obey file to be used by the VARY TCPIP command:

```
ITRACE OFF SUBAGENT
```

For more information on the VARY command, refer to the *z/OS Communications Server: IP Configuration Reference*.

Starting OMPROUTE subagent traces

To start OMPROUTE subagent tracing, start OMPROUTE with the -s1 option or issue the MODIFY SADEBUG command. Output is sent to syslogd. For details, see “Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell” on page 606 and “Starting OMPROUTE tracing and debugging using the MODIFY command” on page 607.

Starting SLA subagent traces

To start SLA subagent tracing, start the SLA subagent with the -d 2 option or by issuing the MODIFY TRACE,LEVEL command. Output is sent to syslogd.

The following are valid SLA subagent trace levels:

- 1 General subagent trace information.
- 2 General subagent trace information, plus extended dump trace and DPI traces. Extended dump trace provides storage dumps of useful information, such as storage returned by the IOCTL calls.

Starting TRAPFWD traces

The following sections provide information about starting TRAPFWD traces.

If TRAPFWD is not running

If TRAPFWD is not already running, specify the -d parameter during startup. You can start the TRAPFWD trace in one of the following ways:

- Through the start options in the JCL used to start the TRAPFWD. For example,


```
//TRAPFWD EXEC PGM=EZASNTRA,REGION=4096K,TIME=NOLIMIT,  
//PARM='POSIX(ON) ALL31(ON)/-d 3'
```

- Through OMVS, using the trapfwd command. For example,
trapfwd -d 3 &

Use one of the following trace levels:

- 1 Minimal tracing, trace address from which the trap is received.
- 2 In addition to 1, trace addresses to which the trap packet is forwarded.
- 3 In addition to 2, trace trap packets.

Trace records are sent to the file specified by the daemon.debug entry in the SYSLOG configuration file. For more information refer to the *z/OS Communications Server: IP Configuration Guide*.

If TRAPFWD is already running

You can use the MVS MODIFY command to start and stop the trace dynamically. Use of this support is restricted to users with MODIFY command privilege.

If you start the trapfwd from JCL, you have no difficulty knowing the procname. However, if you start the trapfwd from OMVS, the trapfwd generates a message to syslogd. This message indicates the job name the trapfwd is running; this is the job name to specify on the MODIFY command.

For example, assume that the procname is TRAPFWD and you want to change the trace level to 3. You would enter the following:

```
MODIFY TRAPFWD,trace,level=3
```

For more information on using the MVS MODIFY command, refer to the *z/OS Communications Server: IP Configuration Reference*.

Trace examples and explanations

The following examples are shown in this section:

- Agent trace
- Subagent traces
- TRAPFWD trace
- NetView SNMP Query Engine trace
- NetView SNMP Query Engine IUCV Communication trace

SNMP agent traces

Figure 55 on page 511 was produced by using **osnmp get sysUpTime.0**. When the SNMP agent is tracing responses, it makes the following entry in the syslogd output file:


```

Dec 19 15:55:38 snmpagent.9.: SNMP logging data follows =====
Dec 19 15:55:39 snmpagent.9.: Log_type:      snmpLOGresponse_out
Dec 19 15:55:39 snmpagent.9.:   send rc:      0
Dec 19 15:55:39 snmpagent.9.:   destination: UDP 127.0.0.1 port 5000
Dec 19 15:55:39 snmpagent.9.:   version:      SNMPv1
Dec 19 15:55:39 snmpagent.9.:   community:    public
Dec 19 15:55:39 snmpagent.9.:   ('70 75 62 6c 69 63'h)
Dec 19 15:55:39 snmpagent.9.:   addressInfo:  UDP 127.0.0.1 port 5000
Dec 19 15:55:39 snmpagent.9.:   PDUtype:      GetResponse ('a2'h)
Dec 19 15:55:39 snmpagent.9.:   request:      1
Dec 19 15:55:39 snmpagent.9.:   error-status: noError (0)
Dec 19 15:55:39 snmpagent.9.:   error-index:  0
Dec 19 15:55:39 snmpagent.9.:   varBind oid:
Dec 19 15:55:39 snmpagent.9.:   OBJECT_IDENTIFIER
Dec 19 15:55:39 snmpagent.9.:   1.3.6.1.2.1.1.3.0
Dec 19 15:55:39 snmpagent.9.:   name:      sysUpTime.0
Dec 19 15:55:39 snmpagent.9.:   value:
Dec 19 15:55:39 snmpagent.9.:   TimeTicks
Dec 19 15:55:39 snmpagent.9.:   5900 - 59.00 seconds
Dec 19 15:55:39 snmpagent.9.: End of SNMP logging data:

```

Figure 55. SNMP agent response trace

In the following scenario, the SNMP agent attempted to initialize, but it was not successful. The port it was using was already in use. The trace shown in Figure 56 was obtained with SNMP agent tracing set to 7.

```

Dec 19 11:57:52 snmpagent.16777227.: EZZ6235I socket function failed for
SNMP inet udp socket; EDC5112I Resource temporarily unavailable.
Dec 19 11:57:52 snmpagent.16777227.: ... errno = 112, errno2 =12fc0296

```

Figure 56. SNMP agent trace of unsuccessful initialization

Note: Errno 112 translates to “Resource temporarily unavailable.” The errno is used primarily by IBM service in diagnosing the error. In this case, issue the `onetstat -c` command to determine if TCP/IP is running and, if so, which ports are in use.

Figure 57 shows the trace produced for the agent when the authentication key sent by a manager does not match the key defined at the agent. The command will receive a report indicating `usmStatsWrongDigests`.

```

IDSTMVS.S@AU1104.SOURCE.S@AGV123(1624): rc=-65 (SNMP_RC_USM_WRONG_DIGEST)
      from snmp_process_message()

```

Figure 57. SNMP messages and agent trace for nonmatching key

Figure 58 on page 512 shows the output received by an SNMP manager and the trace produced for the agent when the operator attempted to retrieve data not within the defined view. The command issued in the z/OS UNIX shell was:

```

osnmp -h v374a -v get usmUserStatus.12.0.0.0.2.0.0.0.0.9.67.35.37.2.117.49

```

```

OUTPUT RECEIVED BY THE MANAGER
usmUserStatus.12.0.0.0.2.0.0.0.9.67.35.37.2.117.49 = noSuchObject

AGENT TRACE
IDSTMVS.S@AU1104.SOURCE.S@AGV123(1624): RC=-30 (SNMP_RC_NOT_IN_VIEW)
from snmp_process_message()

```

Figure 58. SNMP messages and agent trace when data not in defined view

Subagent trace

When requests for MIB variables supported by the TCP/IP subagent fail with an indication that the variable is not supported (noSuchName or noSuchObject), one possibility is that the TCP/IP subagent was unable to connect to the SNMP agent.

Figure 59 illustrates a scenario where the subagent is unable to connect because the password it is using is not accepted by the SNMP agent. (The password used by the subagent is defined or defaulted on the SACONFIG statement in the TCP/IP profile.) The following traces were obtained with SNMP agent traces set to 15 and the subagent traces (as set on the ITRACE profile statement) set to 3.

```

Apr  4 16:28:17 MVS097 snmpagent[67108869]: EZZ6225I SNMP agent: Initialization complete
Apr  4 16:28:21 MVS097 M2SubA[50331651]: VS.2575 do_connect_and_open: DPlconnect_to_agent_UNIXstream rc -2.
Apr  4 16:28:21 MVS097 M2SubA[50331651]: VS.1320 do_open_and_register: Connect to SNMP agent failed, will
keep trying
Apr  4 16:28:21 MVS097 M2SubA[50331651]: VS.1340 do_open_and_register: issue selectex, interval = 10 seconds
Apr  4 16:28:31 MVS097 M2SubA[50331651]: VS.2543 do_connect_and_open .... getting agent info from config
Apr  4 16:28:31 MVS097 M2SubA[50331651]: 08B7A4A0 82818497 A6404040 40404040 40404040 *badpw *
Apr  4 16:28:31 MVS097 M2SubA[50331651]: 08B7A4B0 40404040 40404040 40404040 40404040 * *
Apr  4 16:28:31 MVS097 M2SubA[50331651]: 08B7A49C 000000A1 *.... *
Apr  4 16:28:31 MVS097 M2SubA[50331651]: VS.2556 do_connect_and_open: port 161
Apr  4 16:28:31 MVS097 M2SubA[50331651]: VS.2567 do_connect_and_open: hostname_p => 9.67.35.37
Apr  4 16:28:31 MVS097 snmpagent[67108869]: IDSTMVS.S0064350.SOURCE.S@AGV123(1623): rc=-14
(SNMP_RC_NOT_AUTHENTICATED) from snmp_process_message()
Apr  4 16:28:34 MVS097 snmpagent[67108869]: IDSTMVS.S0064350.SOURCE.S@AGV123(1623): rc=-14
(SNMP_RC_NOT_AUTHENTICATED) from snmp_process_message()

```

Figure 59. SNMP subagent trace

SNMP query engine trace

This section discusses the output produced by the SNMP query engine trace.

Figure 60 on page 514 shows an example of the output produced by the SNMP query engine trace. This trace was produced by starting the SNMP query engine address space with start option -d 4, which is the maximum amount of trace records produced. In the figure, the column labeled “trc lvl” shows the lowest trace level required to see that particular trace entry. For example, lines five through nine have a “trc lvl” of four. This means that only the -d 4 trace option will show this type of trace entry. On the other hand, lines 10 through 17 have a “trc lvl” of two. This means that trace level two or higher will produce this trace information.

Note: The column headed “line no.” numbers the trace records for reference in the discussion that follows the figure. Neither the “trc lvl” nor the “line no.” column appear in the actual trace output.

The following sequence of events occurred to create the trace output:

1. Started the SNMP query engine address space

Trace output lines in the range 1–3

2. Started the SNMPIUCV subtask at the NetView host (attempted connection to the query engine when started)
Trace output line 4
3. Issued an SNMP TRAPSON request (request 1001)
Trace output lines in the range 5–27
4. Incoming SNMP Trap-PDU received from SNMP agent
Trace output lines in the range 28–61
5. Issued an SNMP TRAPSOFF request (request 1002)
Trace output lines in the range 62–82
6. Incoming SNMP Trap-PDU received from SNMP agent
Trace output lines in the range 83–104
7. Issued an SNMP GET request (request 1003)
Trace output lines in the range 105–148
8. Received the response to request 1003
Trace output lines in the range 149–191
9. Issued an SNMP GETNEXT request (request 1004)
Trace output lines in the range 192–235
10. Received the response to request 1004
Trace output lines in the range 236–278
11. Issued an SNMP SET request (request 1005)
Trace output lines in the range 279–326
12. Received the response to request 1005
Trace output lines in the range 327–369
13. Issued an SNMP MIBVNAME request (request 1006)
Trace output lines in the range 370–397
14. Issued an SNMP PING request (request 1007)
Trace output lines in the range 398–429
15. Issued an SNMP GET request for a variable name not defined in the *hlq.MIBDESC.DATA* data set (request 1008)
Trace output lines in the range 430–462
16. Stopped the SNMPIUCV subtask of the NetView program
Trace output line 463

```

trc line
lvl no.

3 1 EZA6322I Using 'TCPCS.mibdesc.data' as MIB description file
0 2 EZA6275I SNMP Query Engine running and awaiting queries...
2 3 EZA6276I There are 56 client connections possible
0 4 EZA6290I Accepted new client connection
4 5 EZA6292I Received following NVquery packet:
6 EZA6305I dumping packet of 19 bytes:
7 00 11 01 01 01 02 06 00 00 03 e9 00 00 00 00 00
8 00 00 00
2 9 EZA6359I major version: 1
10 EZA6360I minor version: 1
11 EZA6361I release: 1
12 EZA6363I native set: EBCDIC
13 EZA6364I packet type: TRAP REQUEST
14 EZA6394I filter id: 1001
15 EZA6396I network mask: 0.0.0.0
16 EZA6397I desired network: 0.0.0.0
2 17 EZA6359I major version: 1
18 EZA6360I minor version: 1
19 EZA6361I release: 1
20 EZA6363I native set: EBCDIC
21 EZA6364I packet type: RESPONSE
22 EZA6367I sequence id: 1001
23 EZA6388I major error: 0
24 EZA6389I minor error: 0
25 EZA6390I error index: 0
26 EZA6391I error text len: 9
27 EZA6392I error text: no error
4 28 EZA6301I Received following SNMP_trap packet:
29 EZA6305I dumping packet of 43 bytes:
30 30 29 02 01 00 04 04 4d 56 53 4c a4 1e 06 0a 2b
31 06 01 04 01 02 02 01 02 04 40 04 09 43 72 25 02
32 01 04 02 01 00 43 02 25 80 30 00
3 33 EZA6424I Decoded SNMP PDU :
34 {
35     version version-1,
36     community '4d56534c'H,
37     data {
38         trap {
39             enterprise 1.3.6.1.4.1.2.2.1.2.4,
40             agent-addr {
41                 internet '09437225'H
42             },
43             generic-trap authenticationFailure,
44             specific-trap 0,
45             time-stamp 9600,
46             variable-bindings {}
47         }
48     }
49 }
4 50 EZA6359I major version: 1
51 EZA6360I minor version: 1
52 EZA6361I release: 1
53 EZA6363I native set: EBCDIC

```

Figure 60. SNMP query engine traces (Part 1 of 10)

```

54 EZA6364I packet type:      TRAP
55 EZA6394I  filter id:      1001
56 EZA6395I  agent address:   9.67.114.37
57 EZA6399I  generic trap:    4      ( 0X4 )
58 EZA6400I  specific trap:   0      ( 0X0 )
59 EZA6401I  time stamp:      9600
60 EZA6402I  enterprise len:   22
61 EZA6403I  enterprise:      1.3.6.1.4.1.2.2.1.2.4
4  62 EZA6292I Received following NVquery packet:
63 EZA6305I dumping packet of 15 bytes:
64 00 0d 01 01 01 02 07 00 00 03 ea 00 00 03 e9
2  65 EZA6359I major version:  1
66 EZA6360I minor version:    1
67 EZA6361I release:          1
68 EZA6363I native set:       EBCDIC
69 EZA6364I packet type:      TRAP UN-REQUEST
70 EZA6367I sequence id:      1002
71 EZA6394I  filter id:      1001
2  72 EZA6359I major version:  1
73 EZA6360I minor version:    1
74 EZA6361I release:          1
75 EZA6363I native set:       EBCDIC
76 EZA6364I packet type:      RESPONSE
77 EZA6367I sequence id:      1002
78 EZA6388I major error:      0
79 EZA6389I minor error:      0
80 EZA6390I error index:      0
81 EZA6391I error text len:   9
82 EZA6392I error text:       no error
4  83 EZA6301I Received following SNMP_trap packet:
84 EZA6305I dumping packet of 43 bytes:
85      30 29 02 01 00 04 04 4d 56 53 4c a4 1e 06 0a 2b
86      06 01 04 01 02 02 01 02 04 40 04 09 43 72 25 02
87      01 04 02 01 00 43 02 38 40 30 00
3  88 EZA6424I Decoded SNMP PDU :
89 {
90     version version-1,
91     community '4d56534c'H,
92     data {
93         trap {
94             enterprise 1.3.6.1.4.1.2.2.1.2.4,
95             agent-addr {
96                 internet '09437225'H
97             },
98             generic-trap authenticationFailure,
99             specific-trap 0,
100            time-stamp 14400,

```

Figure 60. SNMP query engine traces (Part 2 of 10)

```

101         variable-bindings {}
102     }
103 }
104 }
4 105 EZA6292I Received following NVquery packet:
106 EZA6305I dumping packet of 42 bytes:
107 00 28 01 01 01 02 01 00 00 03 eb 00 05 d4 e5 e2
108 d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 0a
109 e2 e8 e2 e4 d7 e3 c9 d4 c5 00
2 110 EZA6359I major version: 1
111 EZA6360I minor version: 1
112 EZA6361I release: 1
113 EZA6363I native set: EBCDIC
114 EZA6364I packet type: GET
115 EZA6367I sequence id: 1003
116 EZA6368I hostname len: 5
117 EZA6370I hostname: MVSL
118 EZA6371I community len: 5
119 EZA6373I community: SNMP
120 EZA6374I optional length: 3
121 EZA6375I max. retries: 1
122 EZA6376I initial timeout: 255
123 EZA6377I backoff exponent: 1
124 EZA6380I name length: 16
125 EZA6381I name: 1.3.6.1.2.1.1.3
3 126 EZA6424I Decoded SNMP PDU :
127 {
128     version version-1,
129     community '534e4d50'H,
130     data {
131         get-request {
132             request-id 1,
133             error-status noError,
134             error-index 0,
135             variable-bindings {
136                 {
137                     name 1.3.6.1.2.1.1.3,
138                     value {
139                         simple {
140                             empty {}
141                         }
142                     }
143                 }
144             }
145         }
146     }
147 }
148 EZA6308I sending SNMP request to 9.67.114.37
4 149 EZA6295I Received following SNMP_response packet:
150 EZA6305I dumping packet of 39 bytes:
151 30 25 02 01 00 04 04 53 4e 4d 50 a2 1a 02 01 01
152 02 01 00 02 01 00 30 0f 30 0d 06 07 2b 06 01 02
153 01 01 03 43 02 48 a8
3 154 EZA6424I Decoded SNMP PDU :

```

Figure 60. SNMP query engine traces (Part 3 of 10)

```

155 {
156     version version-1,
157     community '534e4d50'H,
158     data {
159         get-response {
160             request-id 1,
161             error-status noError,
162             error-index 0,
163             variable-bindings {
164                 {
165                     name 1.3.6.1.2.1.1.3,
166                     value {
167                         application-wide {
168                             ticks 18600
169                         }
170                     }
171                 }
172             }
173         }
174     }
175 }
2 176 EZA6359I major version: 1
177 EZA6360I minor version: 1
178 EZA6361I release: 1
179 EZA6363I native set: EBCDIC
180 EZA6364I packet type: RESPONSE
181 EZA6367I sequence id: 1003
182 EZA6388I major error: 0
183 EZA6389I minor error: 0
184 EZA6390I error index: 0
185 EZA6391I error text len: 9
186 EZA6392I error text: no error
187 EZA6380I name length: 16
188 EZA6381I name: 1.3.6.1.2.1.1.3
189 EZA6382I value type: time ticks
190 EZA6384I value length: 4
191 EZA6387I value: 18600
4 192 EZA6292I Received following NVquery packet:
193 EZA6305I dumping packet of 42 bytes:
194     00 28 01 01 01 02 02 00 00 03 ec 00 05 d4 e5 e2
195     d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 0a
196     c9 c6 c4 c5 e2 c3 d9 4b f0 00
2 197 EZA6359I major version: 1
198 EZA6360I minor version: 1
199 EZA6361I release: 1
200 EZA6363I native set: EBCDIC
201 EZA6364I packet type: GET-NEXT
202 EZA6367I sequence id: 1004
203 EZA6368I hostname len: 5
204 EZA6370I hostname: MVSL
205 EZA6371I community len: 5
206 EZA6373I community: SNMP
207 EZA6374I optional length: 3
208 EZA6375I max. retries: 1
209 EZA6376I initial timeout: 255

```

Figure 60. SNMP query engine traces (Part 4 of 10)

```

210 EZA6377I backoff exponent: 1
211 EZA6380I name length: 22
212 EZA6381I name: 1.3.6.1.2.1.2.2.1.2.0
3 213 EZA6424I Decoded SNMP PDU :
214 {
215     version version-1,
216     community '534e4d50'H,
217     data {
218         get-next-request {
219             request-id 2,
220             error-status noError,
221             error-index 0,
222             variable-bindings {
223                 {
224                     name 1.3.6.1.2.1.2.2.1.2.0,
225                     value {
226                         simple {
227                             empty {}
228                         }
229                     }
230                 }
231             }
232         }
233     }
234 }
235 EZA6308I sending SNMP request to 9.67.114.37
4 236 EZA6295I Received following SNMP_response packet:
237 EZA6305I dumping packet of 47 bytes:
238     30 2d 02 01 00 04 04 53 4e 4d 50 a2 22 02 01 02
239     02 01 00 02 01 00 30 17 30 15 06 0a 2b 06 01 02
240     01 02 02 01 02 01 04 07 49 42 4d 20 4c 43 53
3 241 EZA6424I Decoded SNMP PDU :
242 {
243     version version-1,
244     community '534e4d50'H,
245     data {
246         get-response {
247             request-id 2,
248             error-status noError,
249             error-index 0,
250             variable-bindings {
251                 {
252                     name 1.3.6.1.2.1.2.2.1.2.1,
253                     value {
254                         simple {
255                             string '49424d204c4353'H
256                         }
257                     }
258                 }
259             }
260         }
261     }
262 }
2 263 EZA6359I major version: 1
264 EZA6360I minor version: 1
265 EZA6361I release: 1

```

Figure 60. SNMP query engine traces (Part 5 of 10)


```

266 EZA6363I native set:      EBCDIC
267 EZA6364I packet type:    RESPONSE
268 EZA6367I sequence id:     1004
269 EZA6388I major error:     0
270 EZA6389I minor error:     0
271 EZA6390I error index:     0
272 EZA6391I error text len:  9
273 EZA6392I error text:      no error
274 EZA6380I name length:     22
275 EZA6381I name:            1.3.6.1.2.1.2.2.1.2.1
276 EZA6382I value type:      display string
277 EZA6384I value length:    7
278 EZA6385I value:           IBM LCS
4 279 EZA6292I Received following NVquery packet:
280 EZA6305I dumping packet of 57 bytes:
281          00 37 01 01 01 02 03 00 00 03 ed 00 05 d4 e5 e2
282          d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 10
283          c4 d7 c9 e2 c1 d4 d7 d3 c5 d5 e4 d4 c2 c5 d9 00
284          00 00 06 f1 f2 f3 f4 f5 00
2 285 EZA6359I major version:  1
286 EZA6360I minor version:    1
287 EZA6361I release:          1
288 EZA6363I native set:      EBCDIC
289 EZA6364I packet type:     SET
290 EZA6367I sequence id:     1005
291 EZA6368I hostname len:     5
292 EZA6370I hostname:         MVSL
293 EZA6371I community len:    5
294 EZA6373I community:        SNMP
295 EZA6374I optional length:  3
296 EZA6375I max. retries:     1
297 EZA6376I initial timeout:  255
298 EZA6377I backoff exponent: 1
299 EZA6380I name length:     22

```

Figure 60. SNMP query engine traces (Part 6 of 10)

```

300 EZA6381I  name:          1.3.6.1.4.1.2.2.1.4.1
301 EZA6382I  value type:    number
302 EZA6384I  value length:   4
303 EZA6386I  value:         12345
3 304 EZA6424I Decoded SNMP PDU :
305 {
306     version version-1,
307     community '534e4d50'H,
308     data {
309         set-request {
310             request-id 3,
311             error-status noError,
312             error-index 0,
313             variable-bindings {
314                 {
315                     name 1.3.6.1.4.1.2.2.1.4.1,
316                     value {
317                         simple {
318                             number 12345
319                         }
320                     }
321                 }
322             }
323         }
324     }
325 }
326 EZA6308I sending SNMP request to 9.67.114.37
4 327 EZA6295I Received following SNMP response packet:
328 EZA6305I dumping packet of 42 bytes:
329     30 28 02 01 00 04 04 53 4e 4d 50 a2 1d 02 01 03
330     02 01 00 02 01 00 30 12 30 10 06 0a 2b 06 01 04
331     01 02 02 01 04 01 02 02 30 39
3 332 EZA6424I Decoded SNMP PDU :
333 {
334     version version-1,
335     community '534e4d50'H,
336     data {
337         get-response {
338             request-id 3,
339             error-status noError,
340             error-index 0,
341             variable-bindings {
342                 {
343                     name 1.3.6.1.4.1.2.2.1.4.1,
344                     value {
345                         simple {
346                             number 12345

```

Figure 60. SNMP query engine traces (Part 7 of 10)

```

347     }
348   }
349 }
350 }
351 }
352 }
353 }
2 354 EZA6359I major version: 1
355 EZA6360I minor version: 1
356 EZA6361I release: 1
357 EZA6363I native set: EBCDIC
358 EZA6364I packet type: RESPONSE
359 EZA6367I sequence id: 1005
360 EZA6388I major error: 0
361 EZA6389I minor error: 0
362 EZA6390I error index: 0
363 EZA6391I error text len: 9
364 EZA6392I error text: no error
365 EZA6380I name length: 22
366 EZA6381I name: 1.3.6.1.4.1.2.2.1.4.1
367 EZA6382I value type: number
368 EZA6384I value length: 4
369 EZA6386I value: 12345
4 370 EZA6292I Received following NVquery packet:
371 EZA6305I dumping packet of 29 bytes:
372     00 1b 01 01 01 02 08 00 00 03 ee 00 10 f1 4b f3
373     4b f6 4b f1 4b f2 4b f1 4b f1 4b f1 00
2 374 EZA6359I major version: 1
375 EZA6360I minor version: 1
376 EZA6361I release: 1
377 EZA6363I native set: EBCDIC
378 EZA6364I packet type: VAR_NAME
379 EZA6367I sequence id: 1006
380 EZA6405I object id len: 16
381 EZA6406I object id: 1.3.6.1.2.1.1.1
2 382 EZA6359I major version: 1
383 EZA6360I minor version: 1
384 EZA6361I release: 1
385 EZA6363I native set: EBCDIC
386 EZA6364I packet type: RESPONSE
387 EZA6367I sequence id: 1006
388 EZA6388I major error: 0
389 EZA6389I minor error: 0
390 EZA6390I error index: 0
391 EZA6391I error text len: 9
392 EZA6392I error text: no error
393 EZA6380I name length: 16
394 EZA6381I name: 1.3.6.1.2.1.1.1
395 EZA6382I value type: display string
396 EZA6384I value length: 9
397 EZA6385I value: sysDescr
4 398 EZA6292I Received following NVquery packet:
399 EZA6305I dumping packet of 23 bytes:
400     00 15 01 01 01 02 0a 00 00 03 ef 00 05 d4 e5 e2

```

Figure 60. SNMP query engine traces (Part 8 of 10)

```

401          d3 00 00 03 01 ff 01
2 402 EZA6359I major version: 1
403 EZA6360I minor version: 1
404 EZA6361I release: 1
405 EZA6363I native set: EBCDIC
406 EZA6364I packet type: PING REQUEST
407 EZA6367I sequence id: 1007
408 EZA6368I hostname len: 5
409 EZA6370I hostname: MVSL
410 EZA6374I optional length: 3
411 EZA6375I max. retries: 1
412 EZA6376I initial timeout: 255
413 EZA6377I backoff exponent: 1
2 414 EZA6359I major version: 1
415 EZA6360I minor version: 1
416 EZA6361I release: 1
417 EZA6363I native set: EBCDIC
418 EZA6364I packet type: RESPONSE
419 EZA6367I sequence id: 1007
420 EZA6388I major error: 0
421 EZA6389I minor error: 0
422 EZA6390I error index: 0
423 EZA6391I error text len: 9
424 EZA6392I error text: no error
425 EZA6380I name length: 34
426 EZA6381I name: 1.3.6.1.4.1.2.2.1.3.2.9.67.114.37
427 EZA6382I value type: number
428 EZA6384I value length: 4
429 EZA6386I value: 76
4 430 EZA6292I Received following NVquery packet:
431 EZA6305I dumping packet of 39 bytes:
432          00 25 01 01 01 02 01 00 00 03 f0 00 05 d4 e5 e2
433          d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 07
434          c2 c1 c4 e5 c1 d9 00
1 435 EZA6356E error code 7: unknown MIB variable
2 436 EZA6359I major version: 1
437 EZA6360I minor version: 1
438 EZA6361I release: 1
439 EZA6363I native set: EBCDIC
440 EZA6364I packet type: GET
441 EZA6367I sequence id: 1008
442 EZA6368I hostname len: 5
443 EZA6370I hostname: MVSL
444 EZA6371I community len: 5
445 EZA6373I community: SNMP
446 EZA6374I optional length: 3
447 EZA6375I max. retries: 1
448 EZA6376I initial timeout: 255
449 EZA6377I backoff exponent: 1
450 EZA6380I name length: 2
451 EZA6381I name: ?

```

Figure 60. SNMP query engine traces (Part 9 of 10)

```

2 452 EZA6359I major version: 1
    453 EZA6360I minor version: 1
    454 EZA6361I release: 1
    455 EZA6363I native set: EBCDIC
    456 EZA6364I packet type: RESPONSE
    457 EZA6367I sequence id: 1008
    458 EZA6388I major error: 2
    459 EZA6389I minor error: 7
    460 EZA6390I error index: 0
    461 EZA6391I error text len: 17
    462 EZA6392I error text: unknown variable
0 463 EZA6293I Terminated client connection

```

Figure 60. SNMP query engine traces (Part 10 of 10)

The following is an explanation of the traces in Figure 60 on page 514.

- Line 1 is an information message listing the actual name of the data set being used as the *hlq.MIBDESC.DATA* data set.
- Line 2 is an informational message indicating that the SNMP query engine has been successfully started.
- Line 3 is an informational message indicating the number of client connections the query engine will allow. (A client connection is a connection from a program using the query engine protocol to communicate with the SNMP query engine to initiate SNMP requests. For example, the SNMPIUCV subtask of the NetView program is a client connection).
- Line 4 is an information message indicating that the SNMPIUCV subtask of the NetView program has successfully contacted the query engine.
- Lines 5–8 are the encoded packet received from the client (the SNMPIUCV subtask) by the query engine. This particular packet is the TRAPSON request.
- Lines 9–16 are the decoded SNMPIUCV request. The decoded packet indicates that this request is number 1001 (line 14), and was a TRAPSON request (line 13) for network mask 0.0.0.0 (line 15) with the desired network 0.0.0.0 (line 16).
- Lines 17–27 are the response sent back to SNMPIUCV from the query engine. The response (line 21) is to request number 1001 (line 22) and indicates that the TRAPSON request was successful (lines 23–27).
- Line 28 indicates that an SNMP Trap-PDU was received. Lines 29–32 are the actual BER encoded SNMP packet as it was received by the query engine.
- Lines 33–49 are the decoded version of the trap packet reported by lines 28–32.
- Lines 50–61 are the trap information being passed from the query engine up to the SNMPIUCV subtask to be displayed to the NetView operator. This trap is being forwarded to the NetView program because the IP address of the agent sending the trap (line 56), when ANDed with the network mask (line 15) matches the desired network (line 16) of filter number 1001 (line 55) that was set by the TRAPSON request 1001 (line 14) received previously (lines 9–16).
- Lines 62–64 show an incoming query engine packet sent from SNMPIUCV to the query engine.
- Lines 65–71 are the decoded packet received in lines 62–64. This packet is the TRAPSOFF request (line 69). It requests that trap filter 1001 (line 71) be turned off.
- Lines 72–82 are the response from the query engine to the SNMPIUCV subtask. The response indicates that the TRAPSOFF request was completed successfully (lines 78–82).
- Lines 83–87 indicate that another SNMP Trap-PDU was received from an agent.

- Lines 88–104 are the decoded Trap-PDU. Note that following this decoded PDU, there is no indication of the trap being forwarded to SNMPIUCV. This is because the trap filter has been turned off, so the query engine receives the trap but does not forward the information to SNMPIUCV.
- Lines 105–109 indicate another request from SNMPIUCV being received by the query engine.
- Lines 110–125 are the decoded query engine request. The request from SNMPIUCV was to issue a GetRequest-PDU (line 114) to host MVSL (line 117), using community name SNMP (line 119) and requesting variable 1.3.6.1.2.1.1.3 (line 125). Lines 121–123 are the retry information that SNMPIUCV has gotten from the SNMPARMS member of the DSIPARMS data set.
- Lines 126–147 are the decoded SNMP GetRequest-PDU that the query engine has built as a result of the SNMPIUCV request received in lines 110–125. This PDU has been assigned request number 1 (line 132). This number will be used to correlate the response when it is received.
- Line 148 indicates that the encoded SNMP GetRequest-PDU has been sent to the SNMP agent at the specified IP address. This should be the IP address of the host specified in line 117.
- Line 149 indicates that an SNMP GetResponse-PDU was received. Lines 150–153 are the encoded GetResponse-PDU.
- Lines 154–175 are the decoded GetResponse-PDU. This was a GetResponse (line 159) in response to request number 1 (line 160, matches up to the request number in the request, line 132). The request was completed with no errors (lines 161–162). The requested variable 1.3.6.1.2.1.1.3 (line 165) has a value of 18600 timeticks (line 168).
- Lines 176–191 are the query engine response to SNMPIUCV request number 1003 (lines 115 and 181). The response contains the information received from the agent in the GetResponse-PDU in lines 154–175.
- Lines 192–196 are the next query engine protocol requests received from SNMPIUCV by the query engine.
- Lines 197–212 are the decoded version of the query engine request. This is a GetNext request (line 201) to host MVSL (line 204) for variable 1.3.6.1.2.1.2.2.1.2.0 (line 212). The request number associated with this request is 1004 (line 202).
- Lines 213–234 are the decoded SNMP GetRequest-PDU built as a result of the query engine request received in lines 197–212. This GetRequest-PDU is request number 2 (line 219).
- Line 235 indicates that the encoded GetRequest-PDU has been sent to the requested host.
- Lines 236–240 indicate that a GetResponse-PDU has been received.
- Lines 241–262 are the decoded GetResponse-PDU. This is the response to request number 2 (line 247) for variable 1.3.6.1.2.1.2.2.1.2.1 (line 252). The value of the variable is a display string with the ASCII value of X'49424D204C4353' (line 255). The GetNext request completed successfully (lines 248–249).
- Lines 263–278 are the query engine response to SNMPIUCV request 1004 (line 268). The response contains the information from the GetResponse-PDU (lines 241–262). Note that the variable value in line 255 has been converted to the proper display format in line 278.
- Lines 279–284 are the next query engine protocol request from SNMPIUCV to the query engine.

- Lines 285–303 are the decoded query engine request. It is a SET request (line 289) to host MVSL to set variable 1.3.6.1.4.1.2.2.1.4.1 (line 300) to 12345 (line 303). This is request number 1005 (line 290).
- Lines 304–325 are the decoded SNMP SetRequest-PDU built as a result of the request received in lines 285–303. This is request number 3 (line 310).
- Line 326 indicates that the SetRequest-PDU has been sent to the specified host.
- Lines 327–331 indicate that a GetResponse-PDU has been received.
- Lines 332–353 are the decoded GetResponse-PDU. This PDU is the response to the SetRequest-PDU number 3 (line 338). It was completed successfully (lines 339–340) and variable 1.3.6.1.4.1.2.2.1.4.1 (line 343) was set to 12345 (line 346).
- Lines 354–369 are the query engine response to request 1005 (line 359) containing the information received in the GetResponse-PDU received in lines 332–353.
- Lines 370–373 are the next query engine request packet from SNMPIUCV.
- Lines 374–381 are the decoded query engine request. This is a MIBVNAME request (line 378) requesting the name of variable 1.3.6.1.2.1.1.1 (line 381). The request number is 1006 (line 379).
- Lines 382–397 are the query engine response (line 386) to request 1006 (line 387). The request completed successfully (lines 388–392) and the name of variable 1.3.6.1.2.1.1.1 (line 394) is sysDescr (line 397).
- Lines 398–401 are the next query engine request packet from SNMPIUCV.
- Lines 402–413 are the decoded query engine request packet. This is a PING request (line 406) to ping host MVSL (line 409). The request number is 1007 (line 407).
- Lines 414–429 are the query engine response (line 418) to request 1007 (line 419). The PING completed successfully (lines 420–424) and the round-trip response time was 76 milliseconds (line 429). Note that no SNMP PDUs were generated as a result of the PING request. The SNMP query engine uses a raw socket to send a PING to the requested host and SNMP protocols are not involved.
- Lines 430–434 are the next query engine request packet received from SNMPIUCV.
- Line 435 indicates that an error occurred while the query engine was decoding the request packet. The MIB variable name in the request was unknown to the query engine.
- Lines 435–451 are the decoded query engine request. This is a GET request (line 440). The variable name is unknown (line 451). This is request 1008 (line 441).
- Lines 452–462 are the query engine response (line 456) to SNMPIUCV request 1008 (line 457). The request was unsuccessful. The query engine returns major error code 2 (line 458), minor error code 7 (line 459), unknown variable (line 462). Note that no SNMP PDUs were generated since the query engine could not resolve the variable name.
- Line 463 indicates that the client connection (SNMPIUCV) has been terminated. This is the result of the STOP TASK=SNMPIUCV command.

SNMP query engine IUCV communication trace

Figure 61 on page 526 shows an example of the output produced by the IUCV communication trace. This trace was produced by starting the SNMP query engine address space with start option -it.

```

descarray is at 3985ab8, size is 4 bytes
descarray has 50 entries, entry size is 928
iucvdesc is at 32508
Rc=0 on IUCV_CLR to TCPCS , fd=-254, path=0, iprcode=0, ipmsgid=0, iucvname=00032508
  ciucv_data area (ipbfadr2) is at 00000000
Rc=0 on IUCV_SET to TCPCS , fd=-254, path=0, iprcode=0, ipmsgid=0, iucvname=00032508
  ciucv_data area (ipbfadr2) is at 00005480
Rc=0 on IUCV_CONNECT to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=A0000,
iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00005494
  IUCV interrupt from TCPIP, fd=-254, path=1 type=2 (Connection Complete)
sock_request_inet entry parms:
  f=0 d=-254 r1=00000000 rd=0005ddfc rd1=20 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=C, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 000054bc
  IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=0 d=-254 r1=00000000 rd=0005ddfc rd1=20 pdh=0 pdl=0
  rc=0 err=49 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
  f=25 d=3 r1=00000000 rd=0005db2c rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=3, path=1, iprcode=0, ipmsgid=D, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 000054e4
  IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=25 d=3 r1=00000000 rd=0005db2c rd1=16 pdh=0 pdl=0
  rc=3 err=0 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
  f=2 d=3 r1=00000000 rd=0001d0d8 rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=3, path=1, iprcode=0, ipmsgid=E, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 0000550c
  IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=2 d=3 r1=00000000 rd=0001d0d8 rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
sock_request_inet entry parms:
  f=25 d=4 r1=00000000 rd=0005db44 rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=4, path=1, iprcode=0, ipmsgid=F, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00005534
  IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)

```

Figure 61. SNMP IUCV communication traces (Part 1 of 5)


```

sock_request_inet return parms:
  f=25 d=4 r1=00000000 rd=0005db44 rd1=16 pdh=0 pdl=0
  rc=4 err=0 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
  f=2 d=4 r1=00000000 rd=0005da9c rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=4, path=1, iprcode=0, ipmsgid=10, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 0000555c
  IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=2 d=4 r1=00000000 rd=0005da9c rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
sock_request_inet entry parms:
  f=25 d=5 r1=00000000 rd=0005db64 rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=5, path=1, iprcode=0, ipmsgid=11, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00005584
  IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=25 d=5 r1=00000000 rd=0005db64 rd1=16 pdh=0 pdl=0
  rc=5 err=0 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
  f=2 d=5 r1=00000000 rd=0005daa8 rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=5, path=1, iprcode=0, ipmsgid=12, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 000055ac
  IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=2 d=5 r1=00000000 rd=0005daa8 rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
  f=13 d=5 r1=00000000 rd=00000000 rd1=0 pdh=0 pdl=5
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=5, path=1, iprcode=0, ipmsgid=13, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 000055d4
  IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=13 d=5 r1=00000000 rd=00000000 rd1=0 pdh=0 pdl=5
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_SET to , fd=6, path=0, iprcode=0, ipmsgid=0, iucvname=SNMPQE
  ciucv_data area (ipbfadr2) is at 00005480
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
sock_request_inet entry parms:
  f=25 d=7 r1=00000000 rd=0005db2c rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0

```

Figure 61. SNMP IUCV communication traces (Part 2 of 5)

```

Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=7, path=1, iprcode=0, ipmsgid=14, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=7 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=7 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=7 buf (ipbfadr2) is at 000055fc
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=25 d=7 rl=00000000 rd=0005db2c rdl=16 pdh=0 pdl=0
  rc=7 err=0 rpl=00000000 rpb=00000000 rpbl=0
SQEI001 -- SNMP Query Engine running and awaiting queries...
fd=3 in callers rmask
fd=4 in callers rmask
fd=5 in callers rmask
fd=6 in callers rmask
fd=7 in callers rmask
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
in inetselect
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=15, iucvname=00032508
wait ecblst=5dc5c, ecbscount=2
iucvposted=1073741824, waitposted=0, callposted=0
in iucvposted
Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00005624
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
in gotmsgcomp
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
nfd=0, return=1
sock_request_inet entry parms:
  f=16 d=4 rl=00000000 rd=00000000 rdl=0 pdh=0 pdl=0
  rc=0 err=0 rpl=0005ed88 rpb=00000000 rpbl=4120
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=4, path=1, iprcode=0, ipmsgid=16, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 0000564c
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=16 d=4 rl=00000000 rd=00000000 rdl=0 pdh=0 pdl=0
  rc=0 err=0 rpl=0005ed88 rpb=00000000 rpbl=68
fd=3 in callers rmask
fd=4 in callers rmask
fd=5 in callers rmask
fd=6 in callers rmask
fd=7 in callers rmask
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
in inetselect
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=17, iucvname=00032508
wait ecblst=5dc5c, ecbscount=2
iucvposted=1073741824, waitposted=0, callposted=0
in iucvposted

```

Figure 61. SNMP IUCV communication traces (Part 3 of 5)

```

Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00005674
    IUCV interrupt, fd=6, path=2 type=1 (Pending Connection)
iucvcomp is now TRUE
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
in iucvcom && iucvselect
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
Rc=0 on IUCV_PURGE to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=17, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=6 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_ACCEPT to CNMR3X , fd=8, path=2, iprcode=0, ipmsgid=10000, iucvname=SNMPQE
SQEI002 -- Accepted new client connection
fd=3 in callers rmask
fd=4 in callers rmask
fd=5 in callers rmask
fd=6 in callers rmask
fd=7 in callers rmask
fd=8 in callers rmask
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
in inetselect
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=18, iucvname=00032508
wait ecblst=5dc5c, ecbcount=2
iucvposted=1073741824, waitposted=0, callposted=0
in iucvposted
Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 0000569c
    IUCV interrupt, fd=8, path=2 type=3 (Connection Severed)
iucvcomp is now TRUE
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
in iucvcom && iucvselect
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
Rc=0 on IUCV_PURGE to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=18, iucvname=00032508
fd=8 in callers rmask
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
fd=8 iucvselect now TRUE
in iucvselect, iucvnfds=1
Rc=0 on IUCV_SEVER to CNMR3X , fd=8, path=2, iprcode=0, ipmsgid=0, iucvname=SNMPQE
SQEI003 -- Terminated client connection

```

Figure 61. SNMP IUCV communication traces (Part 4 of 5)

```

fd=3 in callers rmask
fd=4 in callers rmask
fd=5 in callers rmask
fd=6 in callers rmask
fd=7 in callers rmask
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
in inetselect
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=19, iucvname=00032508
wait ecblst=5dc5c, ecbcnt=2
iucvposted=0, waitposted=0, callposted=1073741824
callers ECB posted
Rc=0 on IUCV_PURGE to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=19, iucvname=00032508
Rc=0 on IUCV_CLR to , fd=6, path=0, iprcode=0, ipmsgid=0, iucvname=SNMPQE
ciucv_data area (ipbfadr2) is at 00005480
Rc=0 on IUCV_CLR to TCPCS , fd=-254, path=0, iprcode=0, ipmsgid=0, iucvname=00032508
ciucv_data area (ipbfadr2) is at 00000000

```

Figure 61. SNMP IUCV communication traces (Part 5 of 5)

The following sequence of events occurred to create the trace output:

1. Started the SNMP query engine
2. Connected to the query engine from the SNMPIUCV subtask
3. Disconnected the SNMPIUCV subtask from the query engine

TRAPFWD Trace

The trap forwarder daemon uses syslog functions to write out debug information and traces. Diagnostic data is written using "trapfwd" as identifier.

Figure 62 illustrates a TRAPFWD trace.

```

Oct 15 14:06:06 trapfwd[16777250]: EZZ8420I The Trap Forwarder daemon is running as USER17
Oct 15 14:06:06 trapfwd[16777250]: Establishing affinity with the TCPIP stack
Oct 15 14:06:06 trapfwd[16777250]: Issuing setibmopt for TCPCS
Oct 15 14:06:06 trapfwd[16777250]: Checking if TCP/IP stack is enabled
Oct 15 14:06:06 trapfwd[16777250]: Reading the configuration file : /etc/trapfwd.conf
Oct 15 14:06:06 trapfwd[16777250]: Line 1 : 9.67.113.79 2162
Oct 15 14:06:06 trapfwd[16777250]: Added entry with host: 9.67.113.79 port: 2162
Oct 15 14:06:06 trapfwd[16777250]: Line 2 : 9.67.113.79 1062
Oct 15 14:06:06 trapfwd[16777250]: Added entry with host: 9.67.113.79 port: 1062
Oct 15 14:06:06 trapfwd[16777250]: Line 3 : 9.67.113.79 169
Oct 15 14:06:06 trapfwd[16777250]: Added entry with host: 9.67.113.79 port: 169
Oct 15 14:06:06 trapfwd[16777250]: Line 4 : 9.67.113.79 179
Oct 15 14:06:06 trapfwd[16777250]: Added entry with host: 9.67.113.79 port: 179
Oct 15 14:06:06 trapfwd[16777250]: Creating sockets...
Oct 15 14:06:07 trapfwd[16777250]: EZZ8409I TRAPFWD: INITIALIZATION COMPLETE
Oct 15 14:06:07 trapfwd[16777250]: Ready to receive and forward traps....

```

Figure 62. TRAPFWD trace

Chapter 21. Diagnosing Policy Agent problems

The z/OS UNIX Policy Agent (PAGENT) provides administrative control for policies. This chapter provides information and guidance to diagnose Policy Agent problems.

Overview

Policy Agent reads policies defined in a local file, or read by way of the Lightweight Directory Access Protocol (LDAP) from an LDAP server. These policies are then installed in one or more TCP/IP stacks. Policy Agent can be configured to install identical policies to multiple (or all) stacks, or can install different sets of policies to each stack individually. Policy Agent can also monitor its configuration files and the LDAP server periodically for changed policies, and install new or changed policies as changes occur. Policies are of two basic types:

- Quality of Service (QoS)
- Intrusion Detection Services (IDS)

Refer to the *z/OS Communications Server: IP Configuration Guide* for more information on configuring and starting Policy Agent, as well as defining policies.

The following terms must be defined to understand QoS policies:

Quality of Service (QoS)

The overall service that a user or application receives from a network, in terms of throughput, delay, and such

Service Differentiation

The ability of a network to provide different QoS levels to different users or applications based on their needs.

Service Level Agreement (SLA)

A contract in business terms provided by a network service provider that details the QoS that users or applications are expected to receive.

Service Policy

Administrative controls for a network, in order to achieve the QoS promised by a given SLA.

Integrated Services

A type of service that provides end-to-end QoS to an application, using the methodology of resource reservation along the data path from a receiver to a sender.

Differentiated Services

A type of service that provides QoS to broad classes of traffic or users, for example all FTP traffic to a given subnet.

Resource ReSerVation Protocol (RSVP)

A protocol that provides for resource reservation in support of Integrated Services.

QoS policy scopes

QoS policies can be defined with different scopes. The supported scopes are:

DataTraffic

The policy applies to generic data traffic. This type of policy is in support of Differentiated Services.

- RSVP** The policy applies to RSVP data traffic. This type of policy is in support of Integrated Services.
- TR** The policy applies to Traffic Regulation Management. QoS scope TR policies may only be defined in the Policy Agent configuration file and only support the level of TR function first provided in IBM Communications Server for OS/390 V2R10. With z/OS V1R2 Communications Server, the TR function becomes part of the Intrusion Detection Services and is fully supported by IDS TR policies defined in LDAP. For compatibility, QoS scope TR policies encountered in the Policy Agent configuration file are converted internally by Policy Agent to IDS TR policies.

The TCP/IP stack maps TCP, UDP, and RAW traffic to QoS policies based on the selection criteria defined in the policy. Search criteria can include, but are not limited to, items such as source and destination IP addresses and ports, protocol, and interfaces. The mapping of DataTraffic scoped policies occurs at connect time for TCP traffic, and for each packet for UDP and RAW traffic. For UDP and RAW, however, the mappings are cached such that subsequent packets sent to the same destination will use the cached mapping. RSVP scoped policies are only mapped when the RSVP Agent adds a reservation requested by an RSVP application. The mapping is removed when the reservation is removed. See Chapter 22, “Diagnosing RSVP agent problems” on page 561 for more information on the operation of RSVP.

You can see the effect of defined QoS policies in either of the following ways.

- The SLA Subagent can be used to display service policy and mapped application information, as well as to manage and display SLA performance monitoring.
- Using the z/OS UNIX `pasearch`, z/OS UNIX `onetstat`, and TSO `NETSTAT` commands.
 - The `pasearch` command shows defined policies.
 - The `NETSTAT SLAP` or `netstat -j` command shows performance metrics for active QoS policy Rules.
 - The `NETSTAT ALL` or `onetstat -A` command has additional information for each active connection that shows the QoS policy rule name if the connection maps to a QoS policy.

Refer to the *z/OS Communications Server: IP System Administrator's Commands* for more information on the `NETSTAT/onetstat` commands, the `pasearch` command, and the SLA Subagent. See Chapter 23, “Diagnosing intrusion detection problems” on page 575 for more information on IDS policies.

Gathering diagnostic information

Policy Agent writes logging information to a log file. The level of logged information is controlled by the `LogLevel` configuration statement and the `-d` startup option. This information (loglevel and debug level) can also be changed after startup using the `MODIFY` command. For example:

```
MODIFY procname,LOGLEVEL,LEVEL=127
MODIFY procname,DEBUG,LEVEL=2
```

By default, error, console, warning, and event `LogLevel` messages are written. To gather more diagnostic information, you can specify a `LogLevel` value greater than the default and start the Policy Agent with the `-d 1` startup option to set the debug level to 1. This debug level has the side effect of setting the maximum `LogLevel` value as well.

Use the debug levels as follows:

- Use debug level 1 for most debugging, except Sysplex Distributor performance monitor. This debug value gives extra debugging messages and uses the maximum LogLevel for logging.
- Use debug level 2 to verify Policy Agent processing of LDAP objects, or if a problem is suspected in how LDAP objects are defined.
- Use debug level 4 for summary information concerning Sysplex Distributor performance monitor QoS fraction calculations.
- Use debug level 8 for detailed information concerning Sysplex Distributor performance monitor QoS fraction calculations, and additional Sysplex Distributor debugging.

The trace option -t is used to turn on LDAP client library debugging. Use the trace levels as follows:

- Use trace level 0 for no LDAP client library debugging. This is the default.
- Use trace level 1 to turn on LDAP client library debugging. Note that the destination of LDAP client debug messages is **stderr**. This is controlled by the LDAP client library, not Policy Agent. Using trace level 1 turns on the following LDAP DEBUG Options. For details on debug options, refer to *z/OS Security Server LDAP Server Administration and Use*.

- LDAP_DEBUG_TRACE
- LDAP_DEBUG_PACKETS
- LDAP_DEBUG_ARGS
- LDAP_DEBUG_CONNS
- LDAP_DEBUG_BER
- LDAP_DEBUG_FILTER
- LDAP_DEBUG_MESSAGE
- LDAP_DEBUG_STATS
- LDAP_DEBUG_THREAD
- LDAP_DEBUG_PARSE
- LDAP_DEBUG_PERFORMANCE
- LDAP_DEBUG_REFERRAL
- LDAP_DEBUG_ERROR

If you start Policy Agent with the trace option disabled, then the output destination of **stderr** will be closed. You *cannot* turn on the trace option later with the MODIFY command.

Refer to the *z/OS Communications Server: IP Configuration Reference* for details on how to use the LogLevel, debug level, and trace level.

Log output can be directed either to a set of log files or to the syslog daemon (syslogd). This can be accomplished with the -l startup option or the PAGENT_LOG_FILE environment variable. If output is directed to log files, the number and size of the files can be controlled using the PAGENT_LOG_FILE_CONTROL environment variable. This environment variable can be used to extend the size of the log information collected if necessary. For example, if a large LDAP configuration is used with debug level 2, the default log file size and number may not be sufficient to capture all of the information needed. In this case, use the environment variable to increase the number or size, or the number and size, of the log files. Refer to the *z/OS Communications Server: IP Configuration Guide* for more details on using LogLevel, the -d startup option, and the environment variables, as well as the location of the log file.

Certain other information may be useful in diagnosing Policy Agent problems:

- Output from the pasearch command
- Output from the NETSTAT IDS or onetstat -k commands
- Output from the NETSTAT SLAP or onetstat -j commands
- Output from the NETSTAT ALL or onetstat -A commands for active connections mapped to policies
- SNMP output from walks of the SLA subagent MIB tables
- TCP/IP CTRACE output, using the POLICY, INTERNET and IOCTL CTRACE options
- RSVP Agent log output if RSVP scoped policies are defined

Diagnosing Policy Agent problems

Problems with the Policy Agent will generally fall into one of the following categories:

- Initialization
- Policy definition
- LDAP Object Retrieval
- LDAP Object Storage
- Policy Agent/Sysplex Distribution

Initialization problems

If Policy Agent does not complete initialization, run it with the -d 1 startup option and check the log file for error conditions. If Policy Agent fails to initialize, message EZZ8434I is issued to the console. Check the log file for the specific error encountered. Common problems could include:

- Policy Agent started from a user ID without superuser authority.

Policy Agent must be started from a superuser. The symptoms for this are the EZZ8434I message, along with messages in the log file indicating that superuser authority is required and showing an exit code value of 27.

- Policy Agent not authorized to security product.

Policy Agent must be authorized to a security product profile. Refer to the *z/OS Communications Server: IP Configuration Guide* for details on setting up the proper authorization. The symptoms for this are the EZZ8434I message, along with messages in the log file indicating that the user is not authorized to start Policy Agent and showing an exit code value of 18.

- Unable to read configuration file.

The symptoms for this are the EZZ8434I message, along with messages in the log file indicating that the configuration file could not be opened and showing an exit code value of 1.

- Is the correct configuration file specified? Refer to the *z/OS Communications Server: IP Configuration Guide* for the search order used to locate the main configuration file.
- Does the file exist?
- Are the permission bits correctly set for an HFS file?
- The main configuration file may also include other configuration files for specific TCP/IP stacks. Check these files as well if necessary.

Policy definition problems

If you do not see the expected results when defining policies, check the following. Use the `pasearch` command to display policies (active or inactive) known by Policy Agent. This command can be used to check if policies are active or inactive, and whether or not they contain the specifications that were expected.

Notes:

1. Misspelled attributes are treated by Policy Agent as unknown attributes, and are not flagged in error, so it is the same as if the misspelled attribute were not specified at all.
2. Policy rules with complex conditions (using CNF/DNF logic) are processed by Policy Agent to arrive at a "working" set of conditions. These are the only conditions displayed by default using `pasearch` (use the `-o` option to display the original set of conditions as specified).
3. The `pasearch` output displays overall time ranges and time of day ranges in UTC format, as well as the specified time zone, if other than UTC.

You can now dynamically refresh Policy Agent so that it can pick up any changes made, including changes to policies on the LDAP server (or configuration file). Use the `MODIFY procname, REFRESH` command to restart Policy Agent from the beginning of its configuration files. See Chapter 23, "Diagnosing intrusion detection problems" on page 575 for further information on diagnosing IDS policy definition problems.

To check whether QoS policies are being installed and used correctly, use the `NETSTAT` commands. Use the `NETSTAT SLAP` or `onetstat -j` command to display active QoS policy statistics for QoS policies installed in the stack, as opposed to the policies in Policy Agent. The `NETSTAT ALL` or `onetstat -A` command shows which QoS policy rule (if any) is mapped to active connections.

Some of the problems you may encounter include the following:

- Problems with converting version 1 policies to version 2 policies.

There are some semantic differences between version 1 and version 2 policy definitions. Only the version 2 semantics are currently supported, so version 1 policy semantics are converted to version 2 semantics when the policies are processed by Policy Agent. Be aware of the following processing:

- In version 1, source always meant local, while destination always meant remote. In version 2, source and destination mean exactly what they imply. When version 1 policies specify Direction Inbound, the semantics for source and destination are opposite between the two versions. As a result, although the specified source and destination attributes are displayed as they are specified by the `pasearch` command, the attributes are swapped when the policies are installed in the stack. When converting such policies to version 2, be sure to also swap the source and destination attributes when the version 1 Direction is Inbound. The specified interface is also related to Direction. In version 1 only a single interface is specified, while both inbound and outbound interfaces are specified in version 2. When migrating a version 1 policy, be sure to specify an `InboundInterface` for Direction Inbound, and an `OutboundInterface` for Direction Outbound.
- Similarly to the above, when Direction Both is specified in a version 1 policy, two policies are installed in the stack: one for the outbound direction with source and destination attributes intact, and one for the inbound direction with the attributes swapped. When converting version 1 rules with Direction Both specified, be sure to create two version 2 rules, one for each direction. Also, specify `InboundInterface` for the inbound rule and `OutboundInterface` for the outbound rule, if the version 1 rule specified both Interface and Direction Both.

- PolicyScope values exist in both the policy rule and action in version 1, but only in the policy action in version 2. For any policies that specified different PolicyScope values for the rule and the associated action in version 1, the scope values are merged in the policy action. For example, if the rule specified PolicyScope Both, and the associated action specified PolicyScope DataTraffic, the resulting scope value in the policy action is Both. When converting policies with different PolicyScope values, be sure to logically merge the scopes in the version 2 policy action. Any such merge should always result in a PolicyScope value of Both.
- Policy groups or rules are discarded when defined on an LDAP server.
Policy groups and policy rules defined on an LDAP server may refer to other LDAP objects (such as policy actions or time periods). When any referenced object cannot be found on the LDAP server, the referencing object is discarded. Be sure to specify the correct reference Distinguished Names on LDAP objects that reference other objects.
- Policies with complex conditions (using CNF or DNF) are not mapping correctly.
Complex policy conditions using CNF or DNF can be difficult to properly configure. Since some conditions are logically ANDed, the result may be invalid. For example, two or more distinct interfaces cannot be ANDed and still be true. Or two non-overlapping port ranges also cannot be ANDed. Policy Agent tries to detect these types of errors and discard the policy rules with an error message, but there are cases that cannot be detected (for example, logical ANDs between CNF/DNF levels, or when negated conditions are used). In these cases, a policy rule may well be installed that can never be true. Similar problems could occur when ORing conditions. For example, a very broad condition may map much more traffic than was intended, simply because it is one of a set of conditions that is ORed together. Use the pasearch command to display policy rules with complex conditions. By default, the "working" set of conditions is displayed (after Policy Agent has attempted to collapse and summarize the complex conditions). This working set includes the summary of each condition level, as well as the overall "global" summary condition. Use the pasearch -o option to also display the original set of specified conditions. This will help to show how the working set was derived.
- Wrong policy being mapped to traffic.
There may be times when two or more policy rules are logically mapped to the same set of traffic packets. When this happens, the rule with the highest "weight" is selected. The weight depends on two factors. When the policy rule priority is not specified, the weight depends on the number of attributes specified in the policy conditions. When policy rule priority is specified, the weight is the specified priority plus 100, which is always higher than the weight derived from counting the number of attributes. If more than one rule is found with the same weight, the first such rule is selected to be mapped. Be sure to specify priority in policy rules to better control situations where multiple rules map to the same set of traffic.
- Policies not installed in the TCP/IP stack. The symptom for this is an unexpected or missing set of policies displayed by the NETSTAT SLAP or onetstat -j command.
Is the stack in question configured using a TcpImage statement in Policy Agent configuration file? Check that day of week or time of day, or day of week and time of day, specifications are correct. Verify the specified time zone. For time zones other than local time, the specified time periods may not be currently active. Has the stack been started or restarted after Policy Agent was started? If so, check that the temporary file used by the stack to inform Policy Agent of restarts has not been deleted. Refer to the *z/OS Communications Server: IP Configuration Guide* for more details.

- Policies not mapping to the expected traffic. The symptom for this is a blank policy rule name displayed for an active connection using the NETSTAT ALL or onetstat -A command.

If you think data traffic should be mapped to certain policies, but is not, then check to make sure you have specified the selection criteria correctly on the PolicyRule statement for the policy. For example, TCP policies are mapped on a per connection basis, whereas for UDP and RAW, the policy is mapped on a per packet basis. As an example of TCP traffic, consider an ftp GET request from a remote client. The connection request from the client will be mapped as inbound data, while the data flow will be mapped as outbound data. You can use either source or destination fields in the policy rule to map both traffic flows, but the definitions must be consistent with this way of mapping. Check that the policy is not unnecessarily restrictive in its specification of IP addresses and ports. For RSVP scoped policies, remember that the policy is only mapped to data traffic while an RSVP reservation is in effect.

- Timing windows when switching policies based on time.

If policy rules are defined such that different sets of policies are activated at different times (for example at each shift), be aware of nonoverlapping vs. overlapping time specifications.

For example: if Rule1 is active from 00:00 to 07:29, and Rule2 is active from 07:30 to 04:00, there is a one minute interval gap between these 2 rules. Since the minimum time resolution used by Policy Agent is one minute, there will be a period of one minute when neither policy is active.

- Policies defined in an MVS data set are not being installed.

When an MVS data set is used to define policies, be careful that sequence numbers are not part of the file, as these will cause parsing errors. In ISPF, use the NUMBER OFF and/or UNNUM commands to remove sequence numbers.

LDAP object retrieval problems

If you are having problems receiving policies from an LDAP server, run Policy Agent with the -d 1 and/or 2 startup options, and check the following:

- Unable to connect to the LDAP server.

The symptom for this is message EZZ8440I issued to the console. If Policy Agent fails to connect to the LDAP server, check the log file for the specific error encountered. The Policy Agent keeps trying to connect to the server, using a sliding time window (one minute, then at five minute intervals, with the maximum time between connect attempts being 30 minutes). Note that if a backup LDAP server is configured, the EZZ8440I message is only issued if neither the primary or backup server can be connected.

If you receive message EZZ8440I, check the attributes specified on the ReadFromDirectory statement in the configuration file that relate to the LDAP server connection. These include the primary and backup server addresses and ports, the user ID and password, and SSL parameters. Also check that the specified protocol version matches the protocol version being run on the server.

- No objects, or incorrect objects, retrieved from the LDAP server.

The symptoms for this are missing or incorrect policies displayed by the pasearch command, or the NETSTAT SLAP or onetstat -j commands. Check that the schema version specified on the ReadFromDirectory statement in the configuration file matches the version defined on the LDAP server. The different versions are distinguished by the set of supported object classes. Refer to the *z/OS Communications Server: IP Configuration Guide* for supported schema object classes.

- Wrong set of objects retrieved from the LDAP server. The symptoms for this are missing or incorrect policies displayed by the `pasearch` command or the `NETSTAT SLAP` or `onetstat -j` commands.

Check that the search and selection criteria specified on the `ReadFromDirectory` statement in the configuration file are correct. For version 1 policies, verify that the correct `Base` and `SelectedTag` attributes are used. For version 2 and later policies, check the `SearchPolicyBaseDN`, `SearchPolicyGroupKeyword`, and `SearchPolicyRuleKeyword` attributes.

- LDAP DLL not found.

Policy Agent must have access to the LDAP DLL at run time. The symptom for this is that Policy Agent terminates unexpectedly with a `CEEDUMP`. The reason for termination in the `CEEDUMP` indicates that the LDAP DLL (`GLDCLDAP`) was not found. Policy Agent accesses the LDAP DLL using the `LIBPATH` environment variable. Check that the `LIBPATH` environment variable is specified, and that it contains the directory in which the LDAP DLL (`GLDCLDAP`) resides. This is normally `/usr/lib`.

- Version 1 policies not shared among multiple TCP/IP stacks.

Policy Agent uses two attributes when it searches an LDAP server for version 1 policies that apply to a given TCP/IP image. One attribute is the TCP/IP image name and the other is a selector tag. The selector tag attribute can be defined such that LDAP will scope the search. The TCP/IP image name attribute is set by default to scope the search for a particular image.

Each of the two attributes (`TCPIImageName` and `SelectorTag`) is a multivalue field, meaning you can specify `TCPIImageName/SelectorTag` multiple times in one object defined to LDAP. Both multiple MVS images and multiple TCP/IP stacks can exist. If a policy object is to be used in multiple MVS LPARs, that object can have multiple `SelectorTag` attributes defined, one for each LPAR. If a policy object is to be used in multiple TCP/IP images, that object can have multiple `TCPIImageName` attributes defined, one for each image.

LDAP object storage problems

Policies can be defined on an LDAP server using the appropriate definitions, known as schemas. The policies are defined as object classes with certain attributes, which are a superset of the attributes that can be defined in a local file using the `PolicyAction` and `PolicyRule` statements. Policy Agent acts as an LDAP client to communicate with and retrieve policies from an LDAP server. Policy Agent uses an LDAP DLL to perform its LDAP client functions.

If you are having problems either initializing the LDAP server with the Policy Agent schema definitions, or adding policy objects to the server, check the following:

- Unable to initialize an LDAPv2 server with the Policy Agent schema definition files.

The symptoms may include error messages issued by the server. Since server implementations are different, check the documentation for your server for the types and locations of error or log messages. The Policy Agent LDAPv2 schema definition files are shipped as 2 sample files:

- `/usr/lpp/tcpip/samples/pagent_at.conf`
- `/usr/lpp/tcpip/samples/pagent_oc.conf`

These files need to be included in the LDAP server's initial configuration file, for example `slapd.conf`. Check that these files are included using *include* statements like the following:

```
include /path/pagent_at.conf
include /path/pagent_oc.conf
```

where */path/* defines the location of the schema definition files on the LDAP server. Verify that the files are readable text files.

- Unable to add the Policy Agent schema definitions to an LDAPv3 server.
The symptoms may include error messages issued by the server. Since server implementations are different, check the documentation for your server for the types and locations of error or log messages. The Policy Agent LDAPv3 schema definition files are shipped as 4 sample files:
 - /usr/lpp/tcpip/samples/pagent_schema.ldif
 - /usr/lpp/tcpip/samples/pagent_v3schema.ldif
 - /usr/lpp/tcpip/samples/pagent_schema_updates.ldif
 - /usr/lpp/tcpip/samples/pagent_idsschema.ldif

The *pagent_schema.ldif* file describes the OS/390 V2R10 core/QoS schemas. This file is the base for additional schemas. The *pagent_v3schema.ldif* file describes additions to the core/QoS schema for schema version 3. The *pagent_schema_updates.ldif* file describes updates to the *pagent_schema.ldif* schemas made for z/OS V1R2. Finally, the *pagent_idsschema.ldif* file describes the IDS schema. All of these files need to be installed on the LDAP server in the proper order as an object in the server's database, rather than as configuration information. This process is known as schema publication. Refer to RFCs 1804 and 2251. The files need to be specified on *ldapmodify* commands to modify the *cn:schema* entry in the server's database, in the order as specified in the list above. For example:

```
ldapmodify -h <ldap host or IP address> -p <ldap port> -D "<admin userid>" -w
<admin password> -f /path/pagent_schema.ldif
ldapmodify ... -f /path/pagent_v3schema.ldif
ldapmodify ... -f /path/pagent_schema_updates.ldif
ldapmodify ... -f /path/pagent_idsschema.ldif
```

where */path/* defines the location of the schema definition files on the system where you issue these commands. Verify that the files are readable text files. Verify that the *<suffix>* value on the first noncomment line of these files has been changed to the suffix value defined for your LDAP server, as explained in the prologs in these files.

Note: If the OS/390 V2R10 schema definition was already installed on the server, then the first *ldapmodify* command listed above should not be issued.

- Unable to add policy objects to an LDAP server.
The symptoms may include error messages issued by the server. Since server implementations are different, check the documentation for your server for the types and locations of error or log messages. A typical error message might indicate "object class violation". There are several possible reasons for an LDAP server rejecting a policy object. Check the following:
 - Are the Policy Agent schema definitions installed on the LDAP server?
See above for more information. If the server does not know about policy attributes or object classes, then it will fail any objects that contain them.
 - Are the proper object classes identified for any attributes you have defined in the object?
For example, the *ibm-policySubtreesAuxContainedSet* attribute is defined for the *ibm-policySubtreesPtrAuxClass* object class. If you define a policy object

with this attribute attached, but do not include the object class value, the server will flag the object as an object class violation:

```
dn:cn=example, ou=policy, o=IBM, c=US
objectclass:ibm-policyGroup
objectclass:ibm-policyRuleContainmentAuxClass
cn:example
ibm-policyGroupName:Example-Group
ibm-policyRulesAuxContainedSet:cn=rule1, cn=example, ou=policy, o=IBM, c=US
ibm-policyRulesAuxContainedSet:cn=rule2, cn=example, ou=policy, o=IBM, c=US
ibm-policyRulesAuxContainedSet:cn=rule3, cn=example, ou=policy, o=IBM, c=US
ibm-policySubtreesAuxContainedSet:cn=time, cn=repository, o=IBM, c=US
ibm-policySubtreesAuxContainedSet:cn=cond, cn=repository, o=IBM, c=US
ibm-policySubtreesAuxContainedSet:cn=act, cn=repository, o=IBM, c=US
To correct this, add the object class to the list of objectclass attribute values:
dn:cn=example, ou=policy, o=IBM, c=US
objectclass:ibm-policyGroup
objectclass:ibm-policyRuleContainmentAuxClass
objectclass:ibm-policySubtreesPtrAuxClass
cn:example
ibm-policyGroupName:Example-Group
ibm-policyRulesAuxContainedSet:cn=rule1, cn=example, ou=policy, o=IBM, c=US
ibm-policyRulesAuxContainedSet:cn=rule2, cn=example, ou=policy, o=IBM, c=US
ibm-policyRulesAuxContainedSet:cn=rule3, cn=example, ou=policy, o=IBM, c=US
ibm-policySubtreesAuxContainedSet:cn=time, cn=repository, o=IBM, c=US
ibm-policySubtreesAuxContainedSet:cn=cond, cn=repository, o=IBM, c=US
ibm-policySubtreesAuxContainedSet:cn=act, cn=repository, o=IBM, c=US
```

- Does the server recognize all of your objects?

The symptoms for this are missing objects when you search the server or errors when adding the objects. Some servers, particularly LDAPv2 implementations, may impose strict syntax rules on ldif files that contain objects. For example:

- Lines that separate objects may need just a single newline character. If the separator lines contain other characters, the following object will be processed as a continuation of the previous object. If the object file was transferred using FTP from a host, character translation may result in characters other than newlines separating objects. These additional characters must be removed.
- There must be no blanks at the ends of lines.

Policy Agent/Sysplex distribution problems

The Policy Agent Sysplex Distributor (SD) performance monitor function can be used to calculate outbound network performance information, such as TCP packet loss and timeout ratios, for applications being distributed to on SD target nodes. The calculated performance information is in the form of QoS weight fractions calculated for each DVIPA/Port service level. The QoS weight fractions are used to adjust the WLM weight: the higher the QoS fraction calculation, the lower the adjusted WLM weight. For more information on QoS fractions, refer to the section about Sysplex Distributor Policy Performance Monitoring Configuration in the *z/OS Communications Server: IP Configuration Guide*.

If you suspect problems with the calculated QoS weight fractions, run Policy Agent with debug level 4 or 8 to check the following:

- For debug level 4, Policy Agent displays a summary calculation for each DVIPA/Port XCF address and service level. The summary information includes the retransmit fraction, connection limit fraction, throughput fraction, and the final QoS fraction that resulted. For example:

```
Calculating for DVIPA: 193.1.1.36, Port: 8000, XCF@: 193.1.1.36, SLName:
'Gold_Service'
Fractions: rexmit: 0, connLimit: 100, thruput: 0 QoS used: 100
```

- For debug level 8, Policy Agent displays the intermediate values used to generate the above fractions. For example:

```
Calculating for DVIPA: 193.1.1.36, Port: 8000, XCF@: 193.1.1.36, SLName:
'Gold_Service'
Retransmit Fraction: 0 (Retransmit Bytes: 544, Timeouts: 1,
Octets Sent: 81362424, Segments Sent: 143194)
Connection limit Fraction: 100 (Max Connections: 3, Active Connections: 3)
Throughput Fraction: 0 (Out Bytes: 81362424, Throughput: 10848,
Conn Throughput: 3616 Profile Rate: 0, Min Rate 2000)
QoS Fraction used : 100
```

Note: If the throughput fraction gets set to 100% for any service level, message EZZ8447I is issued. To see which service levels caused this message to be issued, run Policy Agent with debug level 8 and check the log file.

For more information see “Diagnosing Sysplex Distributor problems” on page 41.

Example log file

Figure 63 on page 543 demonstrates some Policy Agent processing. The following configuration files were used to produce this output:

- Main configuration file:

```
TcpImage          TCPCS2 FLUSH PURGE
TcpImage          TCPCS  /u/user10/tcps.conf 600

LogLevel          511

ReadFromDirectory
{
  LDAP_Server      9.37.80.48
  LDAP_Port        7000
  LDAP_DistinguishedName cn=root, o=IBM, c=US
  LDAP_Password    secret
  LDAP_SchemaVersion 3
  LDAP_ProtocolVersion 2
  LDAP_AbstractPolicy No
  SearchPolicyBasedN cn=advanced, ou=policy, o=ibm, c=us
  SearchPolicyKeyword QOS
  SearchPolicyKeyword Sysplex
  SearchPolicyKeyword Attack
}

PolicyPerfMonitorForSDR      Enable
{
  SamplingInterval      60
  LossRatioAndWeightFr  20 25
  TimeoutRatioAndWeightFr 50 50
  LossMaxWeightFr       95
  TimeoutMaxWeightFr    100
}

SetSubnetPrioTosMask
{
  SubnetAddr      9.37.65.139
  SubnetTosMask    11100000
  PriorityTosMapping 1 11100000 7
  PriorityTosMapping 1 11000000 7
  PriorityTosMapping 2 10100000 6
  PriorityTosMapping 2 10000000 5
  PriorityTosMapping 2 01100000 5
  PriorityTosMapping 3 01000000 3
  PriorityTosMapping 4 00100000 2
  PriorityTosMapping 4 00000000 0
}
```

```

PolicyRule                                diffServ
{
  ProtocolNumberRange                     6
  SourceAddressRange                      200.50.23.11
  SourcePortRange                         20-21
  PolicyActionReference                   tokenbucket
  PolicyRulePriority                       10
  ConditionTimeRange                      20000701000000:20050630235959
  DayOfMonthMask                          11111111111111111111111111111111
  DayOfWeekMask                           01111110
  TimeOfDayRange                          06:00-22:00
}

```

```

PolicyAction                              tokenbucket
{
  PolicyScope                             DataTraffic
  OutgoingTOS                             10000000
  DiffServInProfileRate                   256      # 256 Kbps
  DiffServInProfileTokenBucket            512      # 512 Kbits
  DiffServInProfilePeakRate               512      # 512 Kbps
  DiffServInProfileMaxPacketSize          120      # 120 Kbits
  DiffServOutProfileTransmittedTOSByte    00000000
  DiffServExcessTrafficTreatment          BestEffort
}

```

```

PolicyRule                                TRRule
{
  DestinationPortRange                   10000:10100
  DayOfWeekMask                           01111111
  PolicyActionReference                   TRAction
}

```

```

PolicyAction                              TRAction
{
  PolicyScope                             TR
  TypeActions                             Statistics Log
  TotalConnections                        500
  Percentage                              80
  TimeInterval                            120
  LoggingLevel                            7
}

```

- TCPCS Image config file

```

ReadFromDirectory
{
  LDAP_Server                            9.37.80.48
  LDAP_Port                              7000
  LDAP_DistinguishedName                  cn=root, o=IBM, c=US
  LDAP_Password                           badpw
  LDAP_SchemaVersion                      3
  LDAP_ProtocolVersion                    2
  LDAP_AbstractPolicy                     No
  SearchPolicyBaseDN                      cn=advanced, ou=policy, o=ibm, c=us
  SearchPolicyKeyword                     QOS
  SearchPolicyKeyword                     Sysplex
  SearchPolicyKeyword                     Attack
}

PolicyPerfMonitorForSDR                   Enable
{
  SamplingInterval                        60
  LossRatioAndWeightFr                    20 25
  TimeoutRatioAndWeightFr                 50 50
  LossMaxWeightFr                         95
  TimeoutMaxWeightFr                      100
}

```


- Policy Agent Processing Log:

```

01
03/30 17:16:39 LOG      :.main[99]: EZZ8431I PAGENT STARTING
03/30 17:16:39 INFO     :...log_init[99]: Log File Size = 300, Number of Log Files = 3
02
03/30 17:16:39 TRACE    :.main[99]: Option to update policies based on config file(s) update event is active
03/30 17:16:39 INFO     :.main[99]: Use environment PAGENT_CONFIG_FILE = '/u/user10/diagnose.conf'
03/30 17:16:39 INFO     :.main[99]: Using log level 511
03/30 17:16:39 INFO     :..reg_process[99]: registering process with the system
03/30 17:16:39 INFO     :..reg_process[99]: Attempt registration
03/30 17:16:39 INFO     :..reg_process[99]: return from registration rc=0
03/30 17:16:39 INFO     :...mailslot_create[99]: creating mailslot for timer
03/30 17:16:39 INFO     :...mailbox_register[99]: mailbox allocated for timer
03/30 17:16:39 INFO     :..timer_initialize[99]: TimerQ =a0d8fd0
03/30 17:16:39 INFO     :...mailslot_create[99]: creating mailslot for terminate
03/30 17:16:39 INFO     :..mailbox_register[99]: mailbox allocated for terminate
03/30 17:16:39 INFO     :...mailslot_create[99]: creating mailslot for (broken) pipe
03/30 17:16:39 INFO     :..mailbox_register[99]: mailbox allocated for pipe
03/30 17:16:39 INFO     :...mailslot_create[99]: creating mailslot for process abend
03/30 17:16:39 INFO     :..mailbox_register[99]: mailbox allocated for abend
03/30 17:16:39 INFO     :...mailslot_create[99]: creating mailslot for process abort
03/30 17:16:39 INFO     :..mailbox_register[99]: mailbox allocated for abort
03/30 17:16:39 INFO     :...mailslot_create[99]: creating mailslot for process quit
03/30 17:16:39 INFO     :..mailbox_register[99]: mailbox allocated for quit
03/30 17:16:39 INFO     :..profile_initialize[99]: Main config file thread created, tid A560C0000000002
03/30 17:16:39 INFO     :...mailslot_create[99]: creating mailslot for reconfig
03/30 17:16:39 INFO     :..mailbox_register[99]: mailbox allocated for config
03/30 17:16:39 INFO     :...mailslot_create[99]: creating mailslot for user1
03
03/30 17:16:39 EVENT    :.check_main_config_file[98]: Main config file thread active
03/30 17:16:39 TRACE    :.check_main_config_file[98]: pagentRefresh = 0
03/30 17:16:39 INFO     :...pinit_init_tcpimages[98]: processed TcpImage statement: 'TcpImage
TCPCS2 FLUSH PURGE
'
03/30 17:16:39 INFO     :...pinit_init_tcpimages[98]: processed TcpImage statement: 'TcpImage
TCPCS /u/user10/tcps.conf 600
'
03/30 17:16:39 TRACE    :...pinit_create_tmpFile[98]: Creating temporary working file '/tmp/TCPCS.Pagent.tmp'
03/30 17:16:39 INFO     :..mailbox_register[99]: mailbox allocated for user1
03/30 17:16:39 INFO     :.main[99]: Command handler thread created, tid A56180000000003
03/30 17:16:39 TRACE    :...pinit_create_tmpFile[98]: Creating temporary working file '/tmp/TCPCS2.Pagent.tmp'
03/30 17:16:39 TRACE    :..pinit_init_tcpimages[98]: Main config file refresh interval = 600 seconds
03/30 17:16:39 INFO     :.check_main_config_file[98]: Config processing thread created for image 'TCPCS',
index 0, tid A562400000000004
03/30 17:16:39 INFO     :.check_main_config_file[98]: Config processing thread created for image 'TCPCS2',
index 1, tid A563000000000005
03/30 17:16:39 INFO     :..pAPIinit[99]: PAPI server thread created, tid A563C0000000006
03/30 17:16:39 INFO     :...mailslot_create[99]: creating mailslot for dump
03/30 17:16:39 INFO     :..mailbox_register[99]: mailbox allocated for dump
04
03/30 17:16:39 LOG      :.main[99]: EZZ8432I PAGENT INITIALIZATION COMPLETE
03/30 17:16:39 EVENT    :.listen_thread[96]: PAPI server thread active
03/30 17:16:39 EVENT    :.pagent_mvs_command_handler[97]: Command handler thread active
05
03/30 17:16:39 EVENT    :.check_config_files[95]: Config processing thread active for image 'TCPCS',index 0
03/30 17:16:39 EVENT    :.check_config_files[94]: Config processing thread active for image 'TCPCS2',index 1
03/30 17:16:39 INFO     :.check_main_config_file[98]: Config files update event thread created,
tid A565400000000007
03/30 17:16:39 INFO     :.check_main_config_file[98]: Finished main config file update
03/30 17:16:39 INFO     :...settcpimage[95]: Associate with TCP/IP image name = 'TCPCS'

```

Figure 63. Policy Agent (Part 1 of 14)

```

06
03/30 17:16:39 INFO    ...pinit_fetch_policy_profile[95]: Processing policy config data file:
'/u/user10/tcpcs.conf' for image 'TCPCS'
03/30 17:16:39 INFO    ....processing_Stmt_UseLDAPRules[95]: LDAP processing thread created for image
'TCPCS', index 0, tid A56600000000008
03/30 17:16:39 INFO    ....processing_Stmt_PolicyPerfMonit[95]: processing:  'PolicyPerfMonitorForSDR
Enable
,
03/30 17:16:39 TRACE    ....processing_Stmt_PolicyPerfMonit[95]: Turn ON policy performance monitor
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[95]: Performance sampling interval 60 (sec)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[95]: Loss Ratio specified is 20 (in 1/1000 th)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[95]: Loss Weight Fraction specified is 25 (percent)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[95]: Timeout ratio specified is 50 (in 1/1000 th)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[95]: Timeout Weight Fraction specified is
50 (percent)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[95]: Maximum Retrans Weight Fraction specified
is 95(percent)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[95]: Maximum Timeout Weight Fraction specified
is 100(percent)
07
03/30 17:16:39 INFO    ...pinit_fetch_policy_profile[95]: Finish processing above policy config file
03/30 17:16:39 INFO    ..check_config_files[95]: Performance monitor thread created for image 'TCPCS',
index 0,
tid A567800000000009
03/30 17:16:39 INFO    ....settcpimage[94]: Associate with TCP/IP image name = 'TCPCS2'
08
03/30 17:16:39 INFO    ...FlushAllPolicies[94]: Start first by flushing all policies
03/30 17:16:39 INFO    ...profile_delete_ALL_ServiceClass[94]: Image name: 'TCPCS2'
03/30 17:16:39 OBJERR   ...profile_delete_ALL_ServiceClass[94]: Failed reason code =70
03/30 17:16:39 WARNING: ...profile_delete_ALL_ServiceClass[94]: ... reason is: P_RC_POLICY_NOT_FOUND
03/30 17:16:39 EVENT    ...profile_delete_ALL_ServiceClass[94]: Finished deleting ALL Service Class
in image 1
03/30 17:16:39 INFO    ...profile_delete_ALL_PolicyRule[94]: Image name: 'TCPCS2'
03/30 17:16:39 OBJERR   ...profile_delete_ALL_PolicyRule[94]: Failed reason code =70
03/30 17:16:39 WARNING: ...profile_delete_ALL_PolicyRule[94]: ... reason is: P_RC_POLICY_NOT_FOUND
03/30 17:16:39 EVENT    ...profile_delete_ALL_PolicyRule[94]: Finished deleting ALL Policy Rules
09
03/30 17:16:39 INFO    ...pinit_fetch_policy_profile[94]: Processing policy config data file:
'/u/user10/diagnose.conf' for image 'TCPCS2'
03/30 17:16:39 INFO    ....processing_Stmt_UseLDAPRules[94]: LDAP processing thread created for image
'TCPCS2',
index 1, tid A56840000000000A
03/30 17:16:39 INFO    ....processing_Stmt_PolicyPerfMonit[94]: processing:  'PolicyPerfMonitorForSDR
Enable
,
03/30 17:16:39 TRACE    ....processing_Stmt_PolicyPerfMonit[94]: Turn ON policy performance monitor
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[94]: Performance sampling interval 60 (sec)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[94]: Loss Ratio specified is 20 (in 1/1000 th)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[94]: Loss Weight Fraction specified is 25 (percent)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[94]: Timeout ratio specified is 50 (in 1/1000 th)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[94]: Timeout Weight Fraction specified is
50 (percent)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[94]: Maximum Retrans Weight Fraction specified is
95(percent)
03/30 17:16:39 INFO    ....parse_policy_perf_monitor()[94]: Maximum Timeout Weight Fraction specified is
100(percent)
03/30 17:16:39 INFO    ....processing_Stmt_SetSubnetPrioTo[94]: processing:  'SetSubnetPrioToMask
,
03/30 17:16:39 INFO    ....processing_Stmt_ServicePolicyRu[94]: processing:  'PolicyRule
diffServ
,

```

Figure 63. Policy Agent (Part 2 of 14)

```

03/30 17:16:39 INFO      ....processing Stmt_ServiceCategori[94]: processing: 'PolicyAction tokenbucket
,
03/30 17:16:39 INFO      ....processing Stmt_ServicePolicyRu[94]: processing: 'PolicyRule TRule
,
03/30 17:16:39 INFO      ....processing Stmt_ServiceCategori[94]: processing: 'PolicyAction TRAction
,
03/30 17:16:39 TRACE      ....UpdateSCProfileData[94]: Updating SC profile for caller id: 1
03/30 17:16:39 INFO      .....computePolicyRuleWeight[94]: Weight computed for PR 'diffServ' is: 110
10
03/30 17:16:39 WARNING:.....cross_check_PolicyRule[94]: Rule 'TRRule' has reference to PolicyScope
TR action
'TRAction' - converting to IDS rule/action
03/30 17:16:39 INFO      .....computePolicyRuleWeight[94]: Weight computed for PR 'TRRule' is: 1
03/30 17:16:39 TRACE      ....UpdatePRProfileData[94]: Updating PR profile for caller id: 1
03/30 17:16:39 INFO      .....computePolicyRuleWeight[94]: Weight computed for PR 'diffServ' is: 110
03/30 17:16:39 INFO      .....computePolicyRuleWeight[94]: Weight computed for PR 'TRRule' is: 1
11
03/30 17:16:39 INFO      ....profile_install_A_ServiceClass[94]: Service Class: 'tokenbucket'
03/30 17:16:39 EVENT      ....profile_install_A_ServiceClass[94]: Finished installing Service Class:
'tokenbucket'
03/30 17:16:39 INFO      ....profile_install_A_ServiceClass[94]: Service Class: 'TRAction'
03/30 17:16:39 EVENT      ....profile_install_A_ServiceClass[94]: Finished installing Service Class:
'TRAction'
12
03/30 17:16:39 INFO      .....process_time_condition[94]: PR='diffServ' active, next check in 283 minutes
03/30 17:16:39 INFO      .....timer_set[94]:          1st queued=a0b4fa4
03/30 17:16:39 EVENT      ..policy_perf_monitor[93]: Performance monitor thread active for image 'TCPCS',
index 0
03/30 17:16:39 INFO      ...settcpimage[93]: Associate with TCP/IP image name = 'TCPCS'
03/30 17:16:39 EVENT      ....profile_install_A_PolicyRule[94]: Finished installing policy rule: 'diffServ'
03/30 17:16:39 INFO      .....process_time_condition[94]: PR='TRRule' active, next check in 403 minutes
03/30 17:16:39 INFO      .....timer_set[94]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:16:39 INFO      .....timer_set[94]:          last queued=a0b5a6c
03/30 17:16:39 EVENT      ....profile_install_A_PolicyRule[94]: Finished installing policy rule: 'TRRule'
03/30 17:16:39 TRACE      ....clientEventNotify[94]: ClientList Empty
03/30 17:16:39 INFO      ....profile_install_SubnetPrioTosMa[94]: Image name: 'TCPCS2'
13
03/30 17:16:39 SYSERR     ....profile_install_SubnetPrioTosMa[94]: Ioctl to process SetSubnetPrioTosMask
fails, rc=-1, cComCode=121, err=EDC5121I Invalid argument.
03/30 17:16:39 WARNING:...profile_install_SubnetPrioTosMa[94]: ... reason is: P_RC_BAD_SUBNET
03/30 17:16:39 WARNING:...profile_install_SubnetPrioTosMa[94]: Did not finish the command
14
03/30 17:16:39 INFO      ...pinit_fetch_policy_profile[94]: Finish processing above policy config file
03/30 17:16:39 INFO      ..check_config_files[94]: Performance monitor thread created for image 'TCPCS2',
index 1, tid A5690000000000B
15
03/30 17:16:39 EVENT      ..policy_perf_monitor[92]: Performance monitor thread active for image 'TCPCS2',
index 1
03/30 17:16:39 INFO      ....settcpimage[92]: Associate with TCP/IP image name = 'TCPCS2'
16
03/30 17:16:39 EVENT      ..ReadLdapRules[91]: LDAP processing thread active for image 'TCPCS2', index 1
03/30 17:16:39 INFO      ...settcpimage[91]: Associate with TCP/IP image name = 'TCPCS2'
03/30 17:16:39 EVENT      ..ReadLdapRules[91]: Contacting LDAP server '9.37.80.48' on port 7000
03/30 17:16:39 TRACE      ..ReadLdapRules[91]: Update LDAP Trace Level to 0
17
03/30 17:16:39 EVENT      ..ReadLdapRules[90]: LDAP processing thread active for image 'TCPCS', index 0

```

Figure 63. Policy Agent (Part 3 of 14)

```

18
03/30 17:16:39 EVENT  ::config_files_update_event[89]: Config files update event thread active
03/30 17:16:39 INFO   ::config_files_update_event[89]: IPC message queue id obtained, qid =15
03/30 17:16:39 INFO   ::policy_perf_monitor[93]: qosListener thread created for image 'TCPCS',
index 0, tid A569C000000000C
19
03/30 17:16:39 EVENT  ::qosListener[88]: qosListener thread active for image 'TCPCS', index 0
03/30 17:16:39 INFO   ...settcpimage[88]: Associate with TCP/IP image name = 'TCPCS'
03/30 17:16:39 INFO   ::qosListener[88]: Initialization complete for qosListener
03/30 17:16:39 INFO   ::policy_perf_monitor[92]: qosListener thread created for image 'TCPCS2',
index 1, tid A56A8000000000D
20
03/30 17:16:39 EVENT  ::qosListener[87]: qosListener thread active for image 'TCPCS2', index 1
03/30 17:16:39 INFO   ...settcpimage[87]: Associate with TCP/IP image name = 'TCPCS2'
03/30 17:16:39 INFO   ::qosListener[87]: Initialization complete for qosListener
21
03/30 17:16:39 PROTERR:..ReadLdapRules[91]: Cannot bind to directory server: Can't contact LDAP server
03/30 17:16:39 LOG     ..ReadLdapRules[91]: EZZ8440I PAGENT CANNOT CONNECT TO LDAP SERVER FOR TCPCS2
22
03/30 17:16:39 EVENT  ...profile_install_A_PolicyRule[91]: Policy change state flag is set
03/30 17:16:39 EVENT  ...profile_install_A_PolicyRule[91]: Finished installing policy rule: 'Dummy_Rule'
23
03/30 17:16:39 LOG     ..ReadLdapRules[91]: EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPCS2
03/30 17:16:39 INFO   ..ReadLdapRules[91]: Wait for retrying LDAP Connection = 60 seconds
03/30 17:16:39 INFO   ...settcpimage[90]: Associate with TCP/IP image name = 'TCPCS'
03/30 17:16:39 EVENT  ..ReadLdapRules[90]: Contacting LDAP server '9.37.80.48' on port 7000
03/30 17:16:39 TRACE  ..ReadLdapRules[90]: Update LDAP Trace Level to 0
24
03/30 17:16:40 PROTERR:..ReadLdapRules[90]: Cannot bind to directory server: No such object
03/30 17:16:40 LOG     ..ReadLdapRules[90]: EZZ8440I PAGENT CANNOT CONNECT TO LDAP SERVER FOR TCPCS
03/30 17:16:40 EVENT  ...profile_install_A_PolicyRule[90]: Policy change state flag is set
03/30 17:16:40 EVENT  ...profile_install_A_PolicyRule[90]: Finished installing policy rule: 'Dummy_Rule'
03/30 17:16:40 LOG     ..ReadLdapRules[90]: EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPCS
25
03/30 17:16:40 INFO   ..ReadLdapRules[90]: Wait for retrying LDAP Connection = 60 seconds
26
03/30 17:17:00 EVENT  ::config_files_update_event[89]: File Event notification is (__rfim_event)= 1,
(__rfim_type)= 1
03/30 17:17:00 EVENT  ::config_files_update_event[89]: Image config file '/u/user10/tcps.conf'
has been changed
03/30 17:17:00 INFO   ::config_files_update_event[89]: Cancelling config processing thread for image
'TCPCS', index 0, tid A56240000000004
03/30 17:17:00 INFO   ::config_files_update_event[89]: Cancelling LDAP processing thread for image
'TCPCS', index 0, tid A56600000000008
03/30 17:17:00 INFO   ::config_files_update_event[89]: Cancelling performance monitor thread for image
'TCPCS', index 0, tid A56780000000009
03/30 17:17:00 INFO   ::check_config_files[95]: Thread cleanup completed
03/30 17:17:00 INFO   ..ReadLdapRules[90]: Thread cleanup completed
03/30 17:17:00 INFO   ...pqosmon_cleanup[93]: QoS monitor terminating for image : 0
03/30 17:17:00 INFO   ...pqosmon_cleanup[93]: Cancelling qosListener thread for image 'TCPCS', index 0,
tid A569C0000000000C
03/30 17:17:00 INFO   ...pqoslisn_cleanup[88]: qosListener terminating for image : 0...
03/30 17:17:01 INFO   ::qosListener[88]: Thread cleanup completed
03/30 17:17:01 INFO   ::policy_perf_monitor[93]: Thread cleanup completed
03/30 17:17:02 INFO   ::config_files_update_event[89]: Config processing thread created for image 'TCPCS',
index 0, tid A56240000000000E
03/30 17:17:02 EVENT  ::check_config_files[95]: Config processing thread active for image 'TCPCS', index 0
03/30 17:17:02 INFO   ...settcpimage[95]: Associate with TCP/IP image name = 'TCPCS'
03/30 17:17:02 INFO   ...pinit_fetch_policy_profile[95]: Processing policy config data file:
'/u/user10/tcps.conf' for image 'TCPCS'
03/30 17:17:02 INFO   ...processing_stmt_UseLDAPRules[95]: LDAP processing thread created for image
'TCPCS', index 0, tid A56600000000000F
03/30 17:17:02 INFO   ...processing_stmt_PolicyPerfMonit[95]: processing: 'PolicyPerfMonitorForSDR Enable
'

```

Figure 63. Policy Agent (Part 4 of 14)

```

03/30 17:17:02 TRACE :...processing_Stmt_PolicyPerfMonit[95]: Turn ON policy performance monitor
03/30 17:17:02 INFO :....parse_policy_perf_monitor()[95]: Performance sampling interval 60 (sec)
03/30 17:17:02 INFO :....parse_policy_perf_monitor()[95]: Loss Ratio specified is 20 (in 1/1000 th)
03/30 17:17:02 INFO :....parse_policy_perf_monitor()[95]: Loss Weight Fraction specified is 25 (percent)
03/30 17:17:02 EVENT :.ReadLdapRules[93]: LDAP processing thread active for image 'TCPCS', index 0
03/30 17:17:02 INFO :....parse_policy_perf_monitor()[95]: Timeout ratio specified is 50 (in 1/1000 th)
03/30 17:17:02 INFO :....parse_policy_perf_monitor()[95]: Timeout Weight Fraction specified is
50 (percent)
03/30 17:17:02 INFO :....parse_policy_perf_monitor()[95]: Maximum Retrans Weight Fraction specified is
95(percent)
03/30 17:17:02 INFO :....parse_policy_perf_monitor()[95]: Maximum Timeout Weight Fraction specified is
100(percent)
03/30 17:17:02 INFO :..pinit_fetch_policy_profile[95]: Finish processing above policy config file
03/30 17:17:02 INFO :.check_config_files[95]: Performance monitor thread created for image 'TCPCS',
index 0,
tid A56780000000010
03/30 17:17:02 INFO :...settcpimage[93]: Associate with TCP/IP image name = 'TCPCS'
03/30 17:17:02 EVENT :.ReadLdapRules[93]: Contacting LDAP server '9.37.80.48' on port 7000
03/30 17:17:02 TRACE :.ReadLdapRules[93]: Update LDAP Trace Level to 0
03/30 17:17:02 EVENT :.policy_perf_monitor[90]: Performance monitor thread active for image 'TCPCS',
index 0
03/30 17:17:02 INFO :...settcpimage[90]: Associate with TCP/IP image name = 'TCPCS'
03/30 17:17:02 INFO :.policy_perf_monitor[90]: qosListener thread created for image 'TCPCS',
index 0, tid
A569C0000000011
03/30 17:17:02 EVENT :.qosListener[88]: qosListener thread active for image 'TCPCS', index 0
03/30 17:17:02 INFO :...settcpimage[88]: Associate with TCP/IP image name = 'TCPCS'
03/30 17:17:02 INFO :.qosListener[88]: Initialization complete for qosListener
27
03/30 17:17:02 INFO :.ReadLdapRules[93]: Processing version 3 schema
03/30 17:17:02 INFO :...search_ldap[93]: Searching for policies with base 'cn=advanced,ou=policy
,o=ibm,c=us', filter '(&(objectClass=*)(|(ibm-policyKeywords=QOS)(ibm-policyKeywords=Sysplex)
(ibm-policyKeywords=Attack)))', scope subtree
28
03/30 17:17:02 INFO :...sla_ldap_get_policies[93]: LDAP subtree 'cn=advanced,ou=policy,o=ibm,c=us'
searched
03/30 17:17:02 INFO :...sla_ldap_get_policies[93]: 78 objects returned from LDAP search
03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=QoS, cn=advanced,
ou=policy, o=IBM, c=US'
03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=main, cn=QoS,
cn=advanced, ou=policy, o=IBM, c=US'
03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=sysplex, cn=QoS,
cn=advanced, ou=policy, o=IBM, c=US'
03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=diffserv-rule,
cn=QoS, cn=advanced, ou=policy, o=IBM, c=US'
03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=condassoc1,
cn=diffserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US'

... (lines deleted)

03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=condassoc3-1,
cn=attackperpecho-rule, cn=IDS, cn=advanced, ou=policy, o=IBM, c=US'
03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=condassoc3-2,
cn=attackperpecho-rule, cn=IDS, cn=advanced, ou=policy, o=IBM, c=US'
03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=condassoc3-3,
cn=attackperpecho-rule, cn=IDS, cn=advanced, ou=policy, o=IBM, c=US'
03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=condassoc3-4,
cn=attackperpecho-rule, cn=IDS, cn=advanced, ou=policy, o=IBM, c=US'
03/30 17:17:02 TRACE :...process_ldap_group[93]: Search group contains object 'cn=actassoc1,
cn=attackperpecho-rule, cn=IDS, cn=advanced, ou=policy, o=IBM, c=US'
03/30 17:17:02 INFO :...search_ldap[93]: Searching for policies with base 'cn=time,cn=repository,
o=IBM,c=US', filter '(&(objectClass=*)(|(ibm-policyKeywords=QOS)(ibm-policyKeywords=Sysplex)
(ibm-policyKeywords=Attack)))', scope subtree

```

Figure 63. Policy Agent (Part 5 of 14)


```

03/30 17:17:03 INFO    ...sla_ldap_get_policies[93]: LDAP subtree 'cn=time,cn=repository,o=IBM,c=US'
searched
03/30 17:17:03 WARNING:..sla_ldap_get_policies[93]: No objects returned from LDAP search
03/30 17:17:03 INFO    ...search_ldap[93]: Searching for policies with base 'cn=QoScond,cn=repository,
o=IBM,c=US', filter '(&(objectClass=*)(|(ibm-policyKeywords=QOS)(ibm-policyKeywords=Sysplex)
(ibm-policyKeywords=Attack)))', scope subtree
03/30 17:17:03 INFO    ...sla_ldap_get_policies[93]: LDAP subtree 'cn=QoScond,cn=repository,o=IBM,
c=US' searched
03/30 17:17:03 INFO    ...sla_ldap_get_policies[93]: 15 objects returned from LDAP search
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=routedPort,
cn=QoScond, cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=tftpdPort,
cn=QoScond, cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=ftpdPorts,
cn=QoScond, cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=telnetdPort, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=webPort, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=dnsPort, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=IpProtUDP, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=IpProtTCP, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=IpProtOSPF, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=Host1, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=Host2, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=Host3, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=ClientSubnet1, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=ClientSubnet2, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=abcapplPort, cn=QoScond,
cn=repository, o=IBM, c=US'
03/30 17:17:03 INFO    ...search_ldap[93]: Searching for policies with base 'cn=QoSact,cn=repository,o=IBM,
c=US', filter '(&(objectClass=*)(|(ibm-policyKeywords=QOS)(ibm-policyKeywords=Sysplex)(ibm-policy
Keywords=Attack)))', scope subtree
03/30 17:17:03 INFO    ...sla_ldap_get_policies[93]: LDAP subtree 'cn=QoSact,cn=repository,o=IBM,c=US'
searched
03/30 17:17:03 INFO    ...sla_ldap_get_policies[93]: 10 objects returned from LDAP search
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=networkcontrol,
cn=QoSact, cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=internetnetwork,
cn=QoSact, cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=crit-realtime, cn=QoSact,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=interactive1, cn=QoSact,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=interactive2, cn=QoSact,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=batch1, cn=QoSact,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=batch2, cn=QoSact,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=tokenbucket, cn=QoSact,
cn=repository, o=IBM, c=US'

```

Figure 63. Policy Agent (Part 6 of 14)

```
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=connlimit1K, cn=QoSact,
cn=repository, o=IBM, c=US'
03/30 17:17:03 TRACE    ...process_ldap_group[93]: Search group contains object 'cn=telnetGold, cn=QoSact,
cn=repository, o=IBM, c=US'
```

30

```
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=diffserv-rule,cn=QoS,
cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ....search_ldap[93]: Searching for policies with base 'cn=period1,cn=time,
cn=repository,o=IBM,c=US', filter '(&(objectClass=*))', scope base
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=maxconn-rule,cn=QoS,
cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=maxrate-rule,cn=QoS,
cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=web-catalog-rule,
cn=QoS,cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=web-purchase-rule,
cn=QoS,cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=xyzhighpri-rule,
cn=QoS,cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=intserv-rule,cn=QoS,
cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ....search_ldap[93]: Searching for policies with base 'cn=period2,cn=time,
cn=repository,o=IBM,c=US', filter '(&(objectClass=*))', scope base
03/30 17:17:03 INFO     ....search_ldap[93]: Searching for policies with base 'cn=period3,cn=time,
cn=repository,o=IBM,c=US', filter '(&(objectClass=*))', scope base
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=targtelnet-rule,
cn=QoS,cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=webproxy-rule,
cn=QoS,cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=disttelnet-rule,
cn=QoS,cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ...sla_ldap_get_policies[93]: Processing policy rule 'cn=attackipprot-rule,
cn=IDS,cn=advanced,ou=policy,o=IBM,c=US'
03/30 17:17:03 INFO     ....search_ldap[93]: Searching for policies with base 'cn=condassoc1,
cn=attackipprot-rule,cn=IDS,cn=starter,ou=policy,o=IBM,c=US', filter '(&(objectClass=*))', scope base
03/30 17:17:03 INFO     ....search_ldap[93]: Searching for policies with base 'cn=condassoc1a,
cn=attackipprot-rule,cn=IDS,cn=starter,ou=policy,o=IBM,c=US', filter '(&(objectClass=*))', scope base
03/30 17:17:03 INFO     ....search_ldap[93]: Searching for policies with base 'cn=condassoc1b,
cn=attackipprot-rule,cn=IDS,cn=starter,ou=policy,o=IBM,c=US', filter '(&(objectClass=*))', scope base
03/30 17:17:03 INFO     ....search_ldap[93]: Searching for policies with base 'cn=condassoc1c,
cn=attackipprot-rule,cn=IDS,cn=starter,ou=policy,o=IBM,c=US', filter '(&(objectClass=*))', scope base
03/30 17:17:03 INFO     ....search_ldap[93]: Searching for policies with base 'cn=actassoc1,
cn=attackipprot-rule,cn=IDS,cn=starter,ou=policy,o=IBM,c=US', filter '(&(objectClass=*))', scope base

... (lines deleted)

03/30 17:17:08 INFO     ...sla_ldap_get_policies[93]: Processing policy action 'cn=tokenbucket,cn=QoSact,
cn=repository,o=IBM,c=US'
03/30 17:17:08 INFO     ...sla_ldap_get_policies[93]: Processing policy action 'cn=connlimit1K,cn=QoSact,
cn=repository,o=IBM,c=US'
03/30 17:17:08 INFO     ...sla_ldap_get_policies[93]: Processing policy action 'cn=telnetGold,cn=QoSact,
cn=repository,o=IBM,c=US'
03/30 17:17:08 INFO     ...sla_ldap_get_policies[93]: Processing policy action 'cn=attackact1,cn=IDSact,
cn=repository,o=IBM,c=US'
03/30 17:17:08 INFO     ...sla_ldap_get_policies[93]: Processing policy action 'cn=attackact2,cn=IDSact,
cn=repository,o=IBM,c=US'
```

Figure 63. Policy Agent (Part 7 of 14)

```

03/30 17:17:08 TRACE :...UpdateSCProfileData[93]: Updating SC profile for caller id: 2
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'diffserv-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'maxconn-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'maxrate-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'web-catalog-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'web-purchase-rule' is:
110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'xyzhighpri-rule' is:
110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'intserv-rule' is: 4
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'targetlnet-rule' is:
120
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'webproxy-rule' is: 120
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'disttnet-rule' is: 120
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'AttackIPprotAdv-rule'
is: 107
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'AttackIPOpt-rule' is:
107
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'AttackPerpEcho-rule'
is: 107
03/30 17:17:08 TRACE :...UpdatePRProfileData[93]: Updating PR profile for caller id: 2
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'targetlnet-rule' is: 120
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'webproxy-rule' is: 120
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'disttnet-rule' is: 120
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'diffserv-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'maxconn-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'maxrate-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'web-catalog-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'web-purchase-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'xyzhighpri-rule' is: 110
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'AttackIPprotAdv-rule' is:
107
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'AttackIPOpt-rule' is: 107
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'AttackPerpEcho-rule' is:
107
03/30 17:17:08 INFO :....computePolicyRuleWeight[93]: Weight computed for PR 'intserv-rule' is: 4
31
03/30 17:17:08 INFO :...profile_install_A_ServiceClass[93]: Service Class: 'maxrate-action'
03/30 17:17:08 EVENT :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'maxrate-action'
03/30 17:17:08 INFO :...profile_install_A_ServiceClass[93]: Service Class: 'xyzhighpri-action'
03/30 17:17:08 EVENT :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'xyzhighpri-action'
03/30 17:17:08 INFO :...profile_install_A_ServiceClass[93]: Service Class: 'intserv-action1'
03/30 17:17:08 EVENT :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'intserv-action1'
03/30 17:17:08 INFO :...profile_install_A_ServiceClass[93]: Service Class: 'intserv-action2'
03/30 17:17:08 EVENT :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'intserv-action2'
03/30 17:17:08 INFO :...profile_install_A_ServiceClass[93]: Service Class: 'webproxy-action'
03/30 17:17:08 EVENT :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'webproxy-action'
03/30 17:17:08 INFO :...profile_install_A_ServiceClass[93]: Service Class: 'networkcontrol-action'
03/30 17:17:08 EVENT :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'networkcontrol-action'
03/30 17:17:08 INFO :...profile_install_A_ServiceClass[93]: Service Class: 'internetwork-action'
03/30 17:17:08 EVENT :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'internetwork-action'
03/30 17:17:08 INFO :...profile_install_A_ServiceClass[93]: Service Class: 'crit-realtime-action'
03/30 17:17:08 EVENT :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'crit-realtime-action'
03/30 17:17:08 INFO :...profile_install_A_ServiceClass[93]: Service Class:
'interactive1-action'
03/30 17:17:08 EVENT :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'interactive1-action'

```

Figure 63. Policy Agent (Part 8 of 14)


```

03/30 17:17:08 INFO      :...profile_install_A_ServiceClass[93]: Service Class: 'interactive2-action'
03/30 17:17:08 EVENT    :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'interactive2-action'
03/30 17:17:08 INFO      :...profile_install_A_ServiceClass[93]: Service Class: 'batch1-action'
03/30 17:17:08 EVENT    :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'batch1-action'
03/30 17:17:08 INFO      :...profile_install_A_ServiceClass[93]: Service Class: 'batch2-action'
03/30 17:17:08 EVENT    :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'batch2-action'
03/30 17:17:08 INFO      :...profile_install_A_ServiceClass[93]: Service Class: 'tokenbucket-action'
03/30 17:17:08 EVENT    :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'tokenbucket-action'
03/30 17:17:08 INFO      :...profile_install_A_ServiceClass[93]: Service Class: 'connlimit1K-action'
03/30 17:17:08 EVENT    :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'connlimit1K-action'
03/30 17:17:08 INFO      :...profile_install_A_ServiceClass[93]: Service Class: 'telnetGold-action'
03/30 17:17:08 EVENT    :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'telnetGold-action'
03/30 17:17:08 INFO      :...profile_install_A_ServiceClass[93]: Service Class: 'AttackLog-action'
03/30 17:17:08 EVENT    :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'AttackLog-action'
03/30 17:17:08 INFO      :...profile_install_A_ServiceClass[93]: Service Class: 'AttackLimit-action'
03/30 17:17:08 EVENT    :...profile_install_A_ServiceClass[93]: Finished installing Service Class:
'AttackLimit-action'
32
03/30 17:17:08 INFO      :....process_time_condition[93]: PR='targetlnet-rule' active, next check in
283 minutes
03/30 17:17:08 INFO      :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO      :....timer_set[93]:          middle queued=a32ef24
03/30 17:17:08 EVENT    :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'targetlnet-rule'
03/30 17:17:08 INFO      :....process_time_condition[93]: PR='webproxy-rule' active, next check
in 283 minutes
03/30 17:17:08 INFO      :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO      :....timer_set[93]:          middle queued=a32fe4c
03/30 17:17:08 EVENT    :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'webproxy-rule'
03/30 17:17:08 INFO      :....process_time_condition[93]: PR='disttelnet-rule' active, next check
in 283 minutes
03/30 17:17:08 INFO      :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO      :....timer_set[93]:          middle queued=a330c3c
03/30 17:17:08 EVENT    :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'disttelnet-rule'
03/30 17:17:08 INFO      :....process_time_condition[93]: PR='diffserv-rule' active, next check
in 283 minutes
03/30 17:17:08 INFO      :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO      :....timer_set[93]:          middle queued=a33149c
03/30 17:17:08 EVENT    :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'diffserv-rule'
03/30 17:17:08 INFO      :....process_time_condition[93]: PR='maxconn-rule' active, next check
in 403 minutes
03/30 17:17:08 INFO      :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO      :....timer_set[93]:          last queued=a3143dc
03/30 17:17:08 EVENT    :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'maxconn-rule'
03/30 17:17:08 INFO      :....process_time_condition[93]: PR='maxrate-rule' active, next check
in 403 minutes
03/30 17:17:08 INFO      :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO      :....timer_set[93]:          last queued=a32a3e4
03/30 17:17:08 EVENT    :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'maxrate-rule'
03/30 17:17:08 INFO      :....process_time_condition[93]: PR='web-catalog-rule' active, next check
in 403 minutes

```

Figure 63. Policy Agent (Part 9 of 14)

```

03/30 17:17:08 INFO :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO :....timer_set[93]: last queued=a315bc4
03/30 17:17:08 EVENT :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'web-catalog-rule'
03/30 17:17:08 INFO :....process_time_condition[93]: PR='web-purchase-rule' active, next check
in 403 minutes
03/30 17:17:08 INFO :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO :....timer_set[93]: last queued=a30c6ec
03/30 17:17:08 EVENT :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'web-purchase-rule'
03/30 17:17:08 INFO :....process_time_condition[93]: PR='xyzhighpri-rule' active, next check
in 403 minutes
03/30 17:17:08 INFO :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO :....timer_set[93]: last queued=a32234c
03/30 17:17:08 EVENT :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'xyzhighpri-rule'
03/30 17:17:08 INFO :....process_time_condition[93]: PR='AttackIPprotAdv-rule' active, next
check in 403 minutes
03/30 17:17:08 INFO :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO :....timer_set[93]: last queued=a310c64
03/30 17:17:08 EVENT :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'AttackIPprotAdv-rule'
03/30 17:17:08 INFO :....process_time_condition[93]: PR='AttackIPopt-rule' active, next
check in 403 minutes
03/30 17:17:08 INFO :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO :....timer_set[93]: last queued=a2a805c
03/30 17:17:08 EVENT :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'AttackIPopt-rule'
03/30 17:17:08 INFO :....process_time_condition[93]: PR='AttackPerpEcho-rule' active, next
check in 403 minutes
03/30 17:17:08 INFO :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO :....timer_set[93]: last queued=a2a9d6c
03/30 17:17:08 EVENT :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'AttackPerpEcho-rule'
03/30 17:17:08 INFO :....process_time_condition[93]: PR='intserv-rule' inactive, next check
in 103 minutes
03/30 17:17:08 INFO :....timer_set[93]: put timer on existing Q, Qhead=a0b4fa4
03/30 17:17:08 INFO :....timer_set[93]: new Qhead=a30d714
33
03/30 17:17:08 EVENT :...profile_install_A_PolicyRule[93]: Policy change state flag is set
03/30 17:17:08 EVENT :...profile_install_A_PolicyRule[93]: Finished installing policy rule:
'Dummy Rule'
03/30 17:17:08 TRACE :...clientEventNotify[93]: ClientList Empty
34
03/30 17:17:08 LOG :.ReadLdapRules[93]: EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPCS
03/30 17:17:08 LOG :.ReadLdapRules[93]: EZZ8772I PAGENT LDAP POLICY PROCESSING COMPLETE FOR TCPCS
03/30 17:17:08 INFO :...search_ldap[93]: Search for policies with base 'cn=advanced, ou=policy,
o=ibm, c=us', filter '(&(objectClass=setsubnetpriotosmask))', scope subtree
35
03/30 17:17:39 INFO :...policy_perf_get_sampling_data()[92]: Obtained 2 policy performance data entries
from the stack
03/30 17:17:39 INFO :...pqos_refresh_perf_cache[92]: This is the first time wait till we get more data
03/30 17:17:39 EVENT :...policy_perf_send_msg_to_SD()[92]: Sending 0 default fractions to the stack
03/30 17:17:39 INFO :...pqos_get_dist_xcfs[92]: Obtained 2 distributor xcf addresses from the stack
03/30 17:17:39 INFO :...pqos_get_dist_xcfs[92]: Refreshed distributor xcf cache
03/30 17:17:39 INFO :...settcpimage[91]: Associate with TCP/IP image name = 'TCPCS2'
03/30 17:17:39 EVENT :.ReadLdapRules[91]: Contacting LDAP server '9.37.80.48' on port 7000
03/30 17:17:39 TRACE :.ReadLdapRules[91]: Update LDAP Trace Level to 0
03/30 17:17:39 PROTERR:.ReadLdapRules[91]: Cannot bind to directory server: Can't contact LDAP server
03/30 17:17:39 LOG :.ReadLdapRules[91]: EZZ8440I PAGENT CANNOT CONNECT TO LDAP SERVER FOR TCPCS2
03/30 17:17:39 INFO :.ReadLdapRules[91]: Wait for retrying LDAP Connection = 300 seconds
03/30 17:18:02 INFO :...policy_perf_get_sampling_data()[90]: Obtained 8 policy performance data entries
from the stack
03/30 17:18:02 INFO :...pqos_refresh_perf_cache[90]: This is the first time wait till we get more data
03/30 17:18:02 EVENT :...policy_perf_send_msg_to_SD()[90]: Sending 0 default fractions to the stack

```

Figure 63. Policy Agent (Part 10 of 14)

```

36
03/30 17:18:02 INFO    :..pqos_get_dist_xcfs[90]: Obtained 2 distributor xcf addresses from the stack
03/30 17:18:02 INFO    :..pqos_get_dist_xcfs[90]: Refreshed distributor xcf cache
03/30 17:18:02 INFO    :..policy_perf_monitor[90]: qosCollector thread created for image 'TCPCS', index 0,
tid A56B4000000012
37
03/30 17:18:02 EVENT   :..qosCollector[86]: qosCollector thread active for image 'TCPCS', index 0
03/30 17:18:02 INFO    :...settcpimage[86]: Associate with TCP/IP image name = 'TCPCS'
03/30 17:18:02 INFO    :..pqos_refresh_target_cache[86]: Obtained 3 target stack entries from the stack
03/30 17:18:02 INFO    :.....settcpimage[86]: Associate with TCP/IP image name = 'TCPCS'
38
03/30 17:18:02 INFO    :....pqos_send_handshake_req[86]: Successfully sent a handshake req to qosListener
03/30 17:18:02 EVENT   :..qosListener[88]: Accepted connection from 193.1.1.36
03/30 17:18:02 EVENT   :..pqos_rcv_msg_from_collector[88]: Received request from 193.1.1.36 (16 bytes)
03/30 17:18:02 EVENT   :....pqos_send_msg_to_collector[88]: Sending response to 193.1.1.36 (16 bytes)
03/30 17:18:02 INFO    :....pqos_send_handshake_req[86]: Successfully received a handshake response
03/30 17:18:02 LOG     :...pqos_connect_to_listener[86]: EZZ8449I PAGENT RUNNING ON TCPCS CONNECTED
TO PAGENT RUNNING ON 193.1.1.36:1700
03/30 17:18:02 INFO    :.....settcpimage[86]: Associate with TCP/IP image name = 'TCPCS'
03/30 17:18:02 INFO    :....pqos_send_handshake_req[86]: Successfully sent a handshake req to qosListener
03/30 17:18:02 EVENT   :..qosListener[87]: Accepted connection from 193.1.1.36
03/30 17:18:02 EVENT   :..pqos_rcv_msg_from_collector[87]: Received request from 193.1.1.36 (16 bytes)
03/30 17:18:02 EVENT   :....pqos_send_msg_to_collector[87]: Sending response to 193.1.1.36 (16 bytes)
03/30 17:18:02 INFO    :....pqos_send_handshake_req[86]: Successfully received a handshake response
03/30 17:18:02 LOG     :...pqos_connect_to_listener[86]: EZZ8449I PAGENT RUNNING ON TCPCS CONNECTED TO
PAGENT RUNNING ON 193.1.2.36:1700
39
03/30 17:18:02 INFO    :..pqos_build_qos_info_request[86]: Obtained 1 policy action entries from the stack
03/30 17:18:02 EVENT   :..pqos_get_info_from_listeners[86]: Sending request to 193.1.1.36 (60 bytes)
03/30 17:18:02 EVENT   :..pqos_get_info_from_listeners[86]: Sending request to 193.1.2.36 (60 bytes)
03/30 17:18:02 EVENT   :..pqos_rcv_msg_from_collector[88]: Received request from 193.1.1.36 (60 bytes)
03/30 17:18:02 EVENT   :....pqos_send_msg_to_collector[88]: Sending response to 193.1.1.36 (24 bytes)
03/30 17:18:02 EVENT   :..pqos_rcv_msg_from_collector[87]: Received request from 193.1.1.36 (60 bytes)
03/30 17:18:02 EVENT   :....pqos_send_msg_to_collector[87]: Sending response to 193.1.1.36 (24 bytes)
03/30 17:18:02 EVENT   :....pqos_rcv_msg_from_listener[86]: Received response from 193.1.1.36 (24 bytes)
03/30 17:18:02 WARNING:..pqos_send_frns_to_SD[86]: No Values to send down to the stack at this time
03/30 17:18:02 WARNING:..pqos_send_frns_to_SD[86]: cInCnt : 0 bufferSize : 24
03/30 17:18:02 EVENT   :....pqos_rcv_msg_from_listener[86]: Received response from 193.1.2.36 (24 bytes)
03/30 17:18:02 WARNING:..pqos_send_frns_to_SD[86]: No Values to send down to the stack at this time
03/30 17:18:02 WARNING:..pqos_send_frns_to_SD[86]: cInCnt : 0 bufferSize : 24
03/30 17:18:16 INFO    :..listen_thread[96]: PAAPI client thread created for connection 15, tid
A56CC0000000013
40
03/30 17:18:16 EVENT   :..process_rqsts[85]: PAAPI client thread active for connection 15
03/30 17:18:16 TRACE   :..process_rqsts[85]: Received from PAPI Client (0) Query Type = 1
03/30 17:18:16 INFO    :..ProcessRegistration[85]: Pagent Client Name = ' USER10'
03/30 17:18:16 TRACE   :..ProcessRegistration[85]: Pagent Client is Superuser
03/30 17:18:16 TRACE   :..process_rqsts[85]: Received from PAPI Client (1) Query Type = 10
03/30 17:18:16 INFO    :...checkClientPerm[85]: Pagent Superuser Client Name = 'USER10 '
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'targetelnet-rule'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'telnetGold-action'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'webproxy-rule'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'webproxy-action'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'disttelnet-rule'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'telnetGold-action'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'diffserv-rule'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'tokenbucket-action'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'maxconn-rule'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'connlimit1K-action'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'maxrate-rule'
03/30 17:18:16 INFO    :..paapi_search[85]: Found Match to Policy Entry Name = 'maxrate-action'

```

Figure 63. Policy Agent (Part 11 of 14)

```

03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'web-catalog-rule'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'interactivel-action'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'web-purchase-rule'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'crit-realtime-action'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'xyzhighpri-rule'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'xyzhighpri-action'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'AttackIPprotAdv-rule'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'AttackLog-action'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'AttackIPopt-rule'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'AttackLog-action'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'AttackPerpEcho-rule'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'AttackLimit-action'
03/30 17:18:16 INFO    ...checkClientPerm[85]: Pagent Superuser Client Name = 'USER10 '
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'diffServ'
03/30 17:18:16 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'tokenbucket'
03/30 17:18:17 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'TRRule'
03/30 17:18:17 INFO    ...paapi_search[85]: Found Match to Policy Entry Name = 'TRACTION'
03/30 17:18:17 INFO    ...process_rqsts[85]: socket connection closed by client

41
03/30 17:18:34 INFO    ..pagent_mvcs_command_handler[97]: Received a CIBMODIFY: 'QUERY'
03/30 17:18:34 LOG     ..pagent_mvcs_command_handler[97]: EZZ8444I PAGENT LOG LEVEL 511 DEBUG LEVEL 0
TRACE LEVEL 0
03/30 17:18:39 INFO    ...policy_perf_get_sampling_data()[92]: Obtained 2 policy performance data
entries from the stack
03/30 17:18:39 EVENT   ...policy_perf_send_msg_to_SD()[92]: Sending 2 default fractions to the stack
03/30 17:18:39 INFO    ...pqos_get_dist_xcfs[92]: Obtained 2 distributor xcf addresses from the stack
03/30 17:18:39 INFO    ...pqos_get_dist_xcfs[92]: Refreshed distributor xcf cache

42
03/30 17:18:40 INFO    ..pagent_mvcs_command_handler[97]: Received a CIBMODIFY: 'DEBUG,LEVEL=4'
03/30 17:18:40 INFO    ..pagent_mvcs_command_handler[97]: Modify Debug Level = 4
03/30 17:18:40 LOG     ..pagent_mvcs_command_handler[97]: EZZ8443I PAGENT MODIFY COMMAND ACCEPTED

43
03/30 17:19:02 INFO    ...policy_perf_get_sampling_data()[90]: Obtained 8 policy performance data
entries from the stack
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: QoS fraction calculations started
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Calculating for DVIPA: 203.1.1.36, Port: 8000, XCF@:
193.1.1.36, SLName: ''
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Calculating for DVIPA: 203.1.1.36, Port: 8000, XCF@:
193.1.1.36, SLName: 'xyzhighpri-action'
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Calculating for DVIPA: 203.1.1.36, Port: 8000, XCF@:
193.1.1.36, SLName: 'maxrate-action'
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Calculating for DVIPA: 203.1.1.36, Port: 8000, XCF@:
193.1.1.36, SLName: 'connlimit1K-action'
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Calculating for DVIPA: 203.1.1.36, Port: 8002, XCF@:
193.1.1.36, SLName: ''
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Calculating for DVIPA: 203.1.1.36, Port: 8002, XCF@:
193.1.1.36, SLName: 'xyzhighpri-action'
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:02 LOG     ...pqos_refresh_perf_cache[90]: Calculating for DVIPA: 203.1.1.36, Port: 8002, XCF@:
193.1.1.36, SLName: 'maxrate-action'

```

Figure 63. Policy Agent (Part 12 of 14)


```

03/30 17:19:02 LOG      ...pqos_refresh_perf_cache[90]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:02 LOG      ...pqos_refresh_perf_cache[90]: Calculating for DVIPA: 203.1.1.36, Port: 8002, XCF@:
193.1.1.36, SLName: 'connlimit1K-action'
03/30 17:19:02 LOG      ...pqos_refresh_perf_cache[90]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:02 LOG      ...pqos_refresh_perf_cache[90]: QoS fraction calculations ended, # fractions: 8,
elapsed time: 0 seconds
03/30 17:19:02 EVENT    ...policy_perf_send_msg_to_SD()[90]: Sending 2 default fractions to the stack
03/30 17:19:02 INFO     ...pqos_get_dist_xcfs[90]: Obtained 2 distributor xcf addresses from the stack
03/30 17:19:02 INFO     ...pqos_get_dist_xcfs[90]: Refreshed distributor xcf cache
03/30 17:19:02 INFO     ...pqos_refresh_target_cache[86]: Obtained 3 target stack entries from the stack
03/30 17:19:02 INFO     ...pqos_build_qos_info_request[86]: Obtained 1 policy action entries from the stack
03/30 17:19:02 EVENT    ...pqos_get_info_from_listeners[86]: Sending request to 193.1.1.36 (60 bytes)
03/30 17:19:02 EVENT    ...pqos_get_info_from_listeners[86]: Sending request to 193.1.2.36 (60 bytes)
03/30 17:19:02 EVENT    ...pqos_rcv_msg_from_collector[88]: Received request from 193.1.1.36 (60 bytes)
03/30 17:19:02 EVENT    ...pqos_send_msg_to_collector[88]: Sending response to 193.1.1.36 (24 bytes)
03/30 17:19:02 EVENT    ...pqos_rcv_msg_from_collector[87]: Received request from 193.1.1.36 (60 bytes)
03/30 17:19:02 EVENT    ...pqos_send_msg_to_collector[87]: Sending response to 193.1.1.36 (24 bytes)
03/30 17:19:02 EVENT    ...pqos_rcv_msg_from_listener[86]: Received response from 193.1.1.36 (24 bytes)
03/30 17:19:02 WARNING: ...pqos_send_frns_to_SD[86]: No Values to send down to the stack at this time
03/30 17:19:02 WARNING: ...pqos_send_frns_to_SD[86]: cInCnt: 0 bufferSize: 24
03/30 17:19:02 EVENT    ...pqos_rcv_msg_from_listener[86]: Received response from 193.1.2.36 (24 bytes)
03/30 17:19:02 WARNING: ...pqos_send_frns_to_SD[86]: No Values to send down to the stack at this time
03/30 17:19:02 WARNING: ...pqos_send_frns_to_SD[86]: cInCnt: 0 bufferSize: 24
03/30 17:19:39 INFO     ...policy_perf_get_sampling_data()[92]: Obtained 2 policy performance data entries
from the stack
03/30 17:19:39 LOG      ...pqos_refresh_perf_cache[92]: QoS fraction calculations started
03/30 17:19:39 LOG      ...pqos_refresh_perf_cache[92]: Calculating for DVIPA: 203.1.1.36, Port: 8000, XCF@:
193.1.1.36, SLName: ''
03/30 17:19:39 LOG      ...pqos_refresh_perf_cache[92]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:39 LOG      ...pqos_refresh_perf_cache[92]: Calculating for DVIPA: 203.1.1.36, Port: 8002, XCF@:
193.1.1.36, SLName: ''
03/30 17:19:39 LOG      ...pqos_refresh_perf_cache[92]: Fractions: rexmit: 0, connLimit: 0, thruput: 0 QoS
used: 0
03/30 17:19:39 LOG      ...pqos_refresh_perf_cache[92]: QoS fraction calculations ended, # fractions: 2,
elapsed time: 0 seconds
03/30 17:19:39 EVENT    ...policy_perf_send_msg_to_SD()[92]: Sending 2 default fractions to the stack
03/30 17:19:39 INFO     ...pqos_get_dist_xcfs[92]: Obtained 2 distributor xcf addresses from the stack
03/30 17:19:39 INFO     ...pqos_get_dist_xcfs[92]: Refreshed distributor xcf cache
44
03/30 17:19:47 INFO     ...pagent_mvs_command_handler[97]: Received a CIBSTOP
03/30 17:19:47 EVENT    ...mailslot_sitter[99]: process received signal SIGTERM
03/30 17:19:47 TRACE    ...check_signals[99]: received TERM signal
03/30 17:19:47 LOG      .....terminator[99]: EZZ8445I PAGENT SHUTDOWN IN PROGRESS

45
03/30 17:19:47 LOG      .....terminator[99]: EZZ8774I PAGENT POLICIES NOT PURGED FOR TCPCS

03/30 17:19:47 INFO     .....FlushAllPolicies[99]: Start first by flushing all policies
03/30 17:19:47 INFO     .....settcpimage[99]: Associate with TCP/IP image name = 'TCPSC2'
03/30 17:19:47 INFO     .....profile_delete_ALL_ServiceClass[99]: Image name: 'TCPSC2'
03/30 17:19:47 EVENT    .....profile_delete_ALL_ServiceClass[99]: Finished deleting ALL Service Class in
image 1
03/30 17:19:47 INFO     .....settcpimage[99]: Associate with TCP/IP image name = 'TCPSC2'
03/30 17:19:47 INFO     .....profile_delete_ALL_PolicyRule[99]: Image name: 'TCPSC2'
03/30 17:19:47 EVENT    .....profile_delete_ALL_PolicyRule[99]: Finished deleting ALL Policy Rules
03/30 17:19:47 INFO     .....timer_clear[99]: timer id (pnode) a0b4fa4 is canceled
03/30 17:19:47 INFO     .....timer_clear[99]: timer id (pnode) a0b5a6c is canceled
03/30 17:19:47 INFO     .....settcpimage[99]: Associate with TCP/IP image name = 'TCPSC2'
03/30 17:19:47 EVENT    .....profile_install_A_PolicyRule[99]: Policy change state flag is set
03/30 17:19:47 EVENT    .....profile_install_A_PolicyRule[99]: Finished installing policy rule:
'Dummy_Rule'
03/30 17:19:47 LOG      .....terminator[99]: EZZ8773I PAGENT POLICIES PURGED FOR TCPSC2

```

Figure 63. Policy Agent (Part 13 of 14)

```

46
03/30 17:19:47 INFO :.....pAPIterminate[99]: Cancelling PAPI server thread, tid A563C0000000006
03/30 17:19:47 INFO :.....terminator[99]: Terminating global thread, relative id 1
03/30 17:19:47 INFO :.....terminator[99]: Terminating global thread, relative id 2
03/30 17:19:48 INFO :.....terminator[99]: Terminating global thread, relative id 3
03/30 17:19:48 INFO :.....terminator[99]: Terminating image thread, relative id 0
03/30 17:19:48 INFO :.check_config_files[95]: Thread cleanup completed
03/30 17:19:49 INFO :.....terminator[99]: Terminating image thread, relative id 0
03/30 17:19:49 INFO :.check_config_files[94]: Thread cleanup completed
03/30 17:19:49 INFO :.....terminator[99]: Terminating image thread, relative id 1
03/30 17:19:49 INFO :.ReadLdapRules[93]: Thread cleanup completed
03/30 17:19:50 INFO :.....terminator[99]: Terminating image thread, relative id 1
03/30 17:19:50 INFO :.ReadLdapRules[91]: Thread cleanup completed
03/30 17:19:50 INFO :.....terminator[99]: Terminating image thread, relative id 2
03/30 17:19:50 INFO :..pqosmon_cleanup[90]: QoS monitor terminating for image : 0
03/30 17:19:50 INFO :..pqosmon_cleanup[90]: Cancelling qosListener thread for image 'TCPCS', index 0,
tid A569C0000000011
03/30 17:19:50 INFO :..pqoslisn_cleanup[88]: qosListener terminating for image : 0...
03/30 17:19:50 INFO :.qosListener[88]: Thread cleanup completed

47
03/30 17:19:50 SYSERR :..pqos_recv_msg_from_listener[86]: recv with peek failed, errno EDC8121I
Connection reset., errno2 76650446
03/30 17:19:50 OBJERR :..pqos_get_info_from_listeners[86]: pqos_recv_msg_from_listener failed
03/30 17:19:50 LOG :..pqos_get_info_from_listeners[86]: EZZ8446I TCP CONNECTION FROM PAGENT ON TCPCS
TO PAGENT ON 193.1.1.36:1700 IS NO LONGER ACTIVE

03/30 17:19:50 INFO :..pqosmon_cleanup[90]: Cancelling qosCollector thread for image 'TCPCS', index 0,
tid A56B40000000012
03/30 17:19:50 INFO :..pqoscoll_cleanup[86]: qosCollector Terminating for image : 0...
03/30 17:19:50 INFO :..pqos_get_info_from_listeners[86]: Thread cleanup completed
03/30 17:19:50 INFO :..policy_perf_monitor[90]: Thread cleanup completed
03/30 17:19:50 SYSERR :..pqos_recv_msg_from_collector[87]: recv with peek failed, errno EDC8121I Connection
reset., errno2 76650446
03/30 17:19:50 OBJERR :.qosListener[87]: pqos_recv_msg_from_collector failed. rc : -1
03/30 17:19:50 INFO :.qosListener[87]: Removing entry from the qosCollectorInfo
03/30 17:19:51 INFO :.....terminator[99]: Terminating image thread, relative id 2
03/30 17:19:51 INFO :..pqosmon_cleanup[92]: QoS monitor terminating for image : 1
03/30 17:19:51 INFO :..pqosmon_cleanup[92]: Cancelling qosListener thread for image 'TCPCS2', index 1,
tid A56A8000000000D
03/30 17:19:51 INFO :..pqoslisn_cleanup[87]: qosListener terminating for image : 1...
03/30 17:19:51 INFO :.qosListener[87]: Thread cleanup completed
03/30 17:19:51 INFO :..policy_perf_monitor[92]: Thread cleanup completed
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a30d714 is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a32ef24 is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a32fe4c is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a330c3c is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a33149c is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a3143dc is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a32a3e4 is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a315bc4 is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a30c6ec is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a32234c is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a310c64 is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a2a805c is canceled
03/30 17:19:51 INFO :.....timer_clear[99]: timer id (pnode) a2a9d6c is canceled
03/30 17:19:51 INFO :.....terminator[99]: process terminated with exit code 0

48
03/30 17:19:51 INFO :.....terminator[99]: EZZ8433I PAGENT SHUTDOWN COMPLETE

```

Figure 63. Policy Agent (Part 14 of 14)

01

Policy Agent is started. Note that messages with an "EZZ" prefix are also issued to the system console.

Note: The number in brackets on each line (preceding the colon) denotes the thread ID. However, these IDs get dynamically reused as threads are cancelled and restarted.

02

The startup options and main configuration file being used are reported.

03

The main configuration file thread is started.

04

Initialization is complete. Note that this only means main thread initialization. As other threads start, additional processing is done, such as reading image configuration files.

05

The TCP/IP image configuration file threads are started.

06

Processing is started for the TCPCS image configuration file.

07

Processing is completed for the TCPCS image configuration file. Note that an additional thread is started to communicate with the LDAP server.

08

Since the FLUSH parameter was specified for TCPCS2, the Policy Agent first attempts to delete all policies. However, in this case, no policies were previously defined so the deletion attempt fails, and the error is reported.

09

Processing is started for the TCPCS2 image configuration file. Note that this configuration file contains policy statements, which are read and processed.

10

A policy is detected that specifies PolicyScope TR. This scope value is no longer supported, so the policy is converted into an IDS TR policy.

11

The policy actions are installed into the TCPCS2 stack.

12

The policy rules are evaluated to see if they are active or inactive. Active rules are installed into the TCPCS2 stack.

13

An attempt is made to install the SetSubnetPrioTosMask statement values into the TCPCS2 stack. However, the SubnetAddr field on this statement specifies an interface that is only valid for TCPCS, so the error is reported.

14

Processing is completed for the TCPCS2 image configuration file.

15

The performance monitor thread for TCPCS2 is started. This thread is started when the PolicyPerfMonitorForSDR statement is specified.

16

The thread that communicates with the LDAP server for TCPCS2 is started.

17

The thread that communicates with the LDAP server for TCPCS is started.

- 18** The thread to monitor Policy Agent files for dynamic updates is started. This thread monitors all configuration files and a set of temporary files used to detect stack recycles, since the -i startup option was specified.
- 19** The QoS Listener thread for TCPSC is started. This thread listens for connection requests from QoS Collector threads, which are started for each Policy Agent stack that is defined as a Sysplex Distributor distributing stack, if the PolicyPerfMonitorForSDR statement is specified.
- 20** The QoS Listener thread for TCPSC2 is started.
- 21** Since the ReadFromDirectory statement was specified for TCPSC2, an attempt is made to connect to the LDAP server. However, this stack has no connectivity to the server, and the error is reported.
- 22** Policy Agent waits until all policies (both configuration file and LDAP server) are read, parsed, and installed in the stack before notifying the stack that the policy state should be changed. Policy state is used in the stack as an indicator that policies have been added, changed, or deleted. However, when an error occurs in contacting the LDAP server (as in this case), a state change is sent to the stack to pick up any policies that may have been installed from the configuration file, since it can not be known how long it will take (if ever) to contact the LDAP server. If the server is eventually contacted, another state change is sent to the stack after the LDAP server policies are installed.
- 23** A message is issued to indicate that configuration file processing for TCPSC2 is complete.
- 24** Since the ReadFromDirectory statement was specified for TCPSC, an attempt is made to connect to the LDAP server. However, the server password was specified incorrectly, and the error is reported.
- 25** Policy Agent tries to reconnect to the LDAP server using a sliding time interval that starts at one minute, goes to 5 minutes, and repeats at 5 minute intervals up to 30 minutes.
- 26** The configuration file is corrected to specify the right password. Since the -i startup option was specified, the change is immediately detected. Because no assumptions can be made about the contents of the configuration file relative to its previous contents, Policy Agent cancels all threads associated with TCPSC, and rereads the configuration file. Threads are restarted as appropriate.
- 27** Another attempt is made to connect to the LDAP server. Since the server password was corrected, this connect attempt is successful.
- 28** The LDAP directory subtree and the number of objects returned from the

server are reported. Each object retrieved is reported in the log because the maximum LogLevel is in effect. Some of the log file has been deleted for purpose of brevity.

29

Additional directory subtrees, pointed to by one or more of the initial set of objects returned, are searched. This continues recursively until all such referenced subtrees are retrieved. Note that additional objects may also need to be individually retrieved if any policy groups or rules point to such objects and if they have not been yet retrieved during this initial subtree search loop.

30

The policy rules retrieved from the LDAP server are processed. Again, some of the log file has been deleted for the purpose of brevity.

31

The policy actions read from the LDAP server are installed into the TCPCS stack.

32

The policy rules read from the LDAP server are evaluated to see if they are active or inactive. Active rules are installed into the TCPCS stack.

33

Once all of the LDAP server policies are installed, a state change is sent to the stack.

34

Messages are issued to the console to indicate that configuration file and LDAP server processing are complete for TCPCS.

35

The performance monitor thread for TCPCS2 begins calculations. Since this is the first iteration, there is no work to do. Only when current and previous values have been collected can performance calculations be made. Similar processing occurs for the TCPCS performance monitor thread.

36

The performance monitor thread for TCPCS determines that it is running on a Sysplex Distributor distributing stack.

37

The QoS Collector thread for TCPCS is started.

38

The QoS Collector thread tries to communicate with all QoS Listener threads. For this example, there are two Listener Threads: TCPCS and TCPCS2. Successful connections are established with each of these and reported with console messages.

39

The QoS Collector thread reads all policy actions that are associated with Sysplex Distributor rules (only 1 in this example). It then requests and receives performance information from each QoS Listener thread for this policy action.

40

A pasearch command is processed. This particular command uses the default parameters, meaning that all active rules and all associated actions are returned for the search.

- 41** A MODIFY QUERY command is entered, and a message is issued showing the current values for LogLevel, debug level, and trace level.
- 42** A MODIFY DEBUG command is entered to change the debug level to 4.
- 43** The performance monitor thread for TCPCS begins calculations for the next sampling interval. Since the debug level is now 4, messages are logged showing a summary of each performance calculation.
- 44** A STOP (P) command is entered to shut down Policy Agent.
- 45** The TcpImage statement for TCPCS uses the default NOPURGE parameter, so a message is issued indicating that policies are not purged from this stack. TCPCS2 specifies PURGE, so the policies are deleted from the stack and an appropriate message is issued.
- 46** All threads are cancelled.
- 47** Because of the order of thread termination, the QoS Listener or the QoS Collector threads, or the QoS Listener and the QoS Collector threads, may see errors on the TCP connection between them. These errors are a result of timing during termination, and can be ignored.
- 48** Policy Agent termination is complete.

Chapter 22. Diagnosing RSVP agent problems

The z/OS UNIX RSVP Agent provides end-to-end resource reservation services on behalf of applications. This chapter provides information and guidance to diagnose z/OS UNIX RSVP Agent problems.

Overview

The RSVP Agent provides an RSVP Application Programming Interface (RAPI) for QoS-aware applications to use. Applications use RAPI to register their intent to use RSVP services, to describe their data traffic, and to explicitly request that network resources be reserved on their behalf. The RSVP Agent communicates with its peers (other RSVP Agents running on z/OS or other platforms) in the network, with QoS-aware sender and receiver applications, and with the TCP/IP stack to effect resource reservations. Refer to RFC 2205 for more information on RSVP, and to the *z/OS Communications Server: IP Programmer's Reference* for more information on RAPI.

The following terms must be defined to understand RSVP processing:

Quality of Service (QoS)

The overall service that a user or application receives from a network, in terms of throughput, delay, and such.

QoS-Aware Application

An application that explicitly requests QoS services from the RSVP agent.

Service Differentiation

The ability of a network to provide different levels of QoS to different users or applications based on their needs.

Service Level Agreement (SLA)

A contract in business terms provided by a network service provider that details the QoS that users or applications are expected to receive.

Service Policy

Administrative controls for a network, in order to achieve the QoS promised by a given SLA.

Integrated Services

A type of service that provides end-to-end QoS to an application, using the methodology of resource reservation along the data path from a receiver to a sender.

Differentiated Services

A type of service that provides QoS to broad classes of traffic or users, for example all FTP traffic to a given subnet.

Resource ReSerVation Protocol (RSVP)

A protocol that provides for resource reservation in support of Integrated Services.

Reservation types, styles, and objects

There are two types of Integrated Services reservations used by the RSVP Agent:

Controlled Load

This reservation type is designed to make the network behave as though it were not loaded, even if one or more of the network elements are experiencing a heavy traffic load. Refer to RFC 2211 for more information on this service.

Guaranteed

This reservation type is designed to allow the network to compute the maximum delay data traffic will receive from the network, based on the traffic specification and other known data. Refer to RFC 2212 for more information on this service.

In addition, there are three styles of reservation, depending on how the receiver desires to apply the reservation to its senders:

WF (Wildcard Filter)

This style applies a single reservation request to all senders.

FF (Fixed Filter)

This style pairs a given reservation request to a given sender. In this way, the receiver can apply a different reservation to each of its senders.

SE (Shared Explicit)

This style applies a single reservation to a list of senders. This differs from the WF style in that the list of senders is finite. Additional senders that appear in the future will not automatically inherit an SE style reservation.

Several objects are used in RSVP and RAPI to describe data traffic and reservations. These objects are as follows:

Tspec (traffic specification)

The Tspec is used to describe the sending application data traffic characteristics. It consists of an object known as a token bucket and other related values. A token bucket is a continually sustainable data rate, and the extent to which the rate can exceed the sustainable level for short periods of time. More detail concerning token buckets and other Integrated Services parameters and processing can be found in RFCs 2210, 2211, 2212, and 2215.

The Tspec contains these values:

- r** Token bucket rate, in bytes per second
- b** Token bucket depth, in bytes
- p** Peak rate, in bytes per second
- m** Minimum policed unit (minimum packet size to be considered), in bytes
- M** Maximum packet size (MTU), in bytes

Rspec (guaranteed receiver specification)

An Rspec consists of two values that further describe a reservation request when Guaranteed service is being used:

- R** Requested rate, in bytes per second
- S** Slack term, in microseconds

Flowspec (reservation specification)

The flowspec is the object used by a receiver application to indicate an actual reservation to be made. The actual makeup of the flowspec depends

on the type of reservation. For Controlled Load, the flowspec takes the same form as the sender Tspec (although the form is the same, the receiver may specify different values than the sender). For Guaranteed, the flowspec takes the form of a Tspec followed by an Rspec.

Policies and RSVP processing

Policies can be defined with RSVP scope. The RSVP Agent obtains a service policy for which traffic is mapped (if any) from the Policy Agent when an application using RAPI indicates it is a sender (when the Tspec is first provided), or when it requests a reservation as a receiver (when the Rspec is first provided for Guaranteed service). At both of these times, if a service policy is defined that maps to the data traffic, the RSVP Agent uses values in the service policy to limit the request from the application. Specifically, the following are limited:

- Total number of RSVP flows.

The MaxFlows keyword on the PolicyAction statement of the policy definition can be used to limit the total number of application flows that use RSVP services.

- Tspec token bucket values.

The MaxRatePerFlow and MaxTokenBucketPerFlow keywords on the PolicyAction statement of the policy definition can be used to limit the r and b values, respectively, in the sender supplied Tspec.

- Rspec values.

The MaxRatePerFlow keyword on the PolicyAction statement of the policy definition can be used to limit the R value in the receiver supplied Rspec.

- Reservation type.

The FlowServiceType keyword on the PolicyAction statement of the policy definition can be used to limit the type of reservation requested. A Guaranteed type request is considered to be "greater than" a Controlled Load type request. So if an application requests Guaranteed, but the policy limits the type to Controlled Load, the reservation will use Controlled Load.

RSVP processing proceeds as follows. When an application uses RAPI to indicate it is a sender, the RSVP Agent packages the sender Tspec (along with other information) in an RSVP PATH packet, and sends the packet to the final destination. The packet is sent using RAW sockets, with the IP Router Alert option set. This option causes each router that supports RSVP to intercept the PATH packet, for the purpose of remembering the PATH request, and to insert a "previous hop" object into the packet, which is then sent again to the final destination. This causes the packet to eventually arrive at the destination, with all RSVP routers in the data path aware of the RSVP flow. At the destination, the RSVP Agent passes the PATH packet to the application, using RAPI. The receiver application uses the Tspec and other information to arrive at a reservation request (flowspec). The receiver application uses RAPI to pass this flowspec to the RSVP Agent. The RSVP Agent then sends an RSVP RESV packet (containing the flowspec and other information) to the previous hop. Each router or host along the path back to the sender receives this RESV packet, uses the flowspec to install the appropriate reservation (if possible), and forwards the RESV to its previous hop. In this way, each RSVP-capable router or host along the data path installs the reservation according to its capabilities. At the sender, the RSVP Agent passes the RESV packet information to the sender application, which then has information that indicates the actual reservation in place. The sender may choose to wait for the reservation to be in place, or may begin sending data before this happens (although such data will of course be treated by the network as though no reservation were in place). Any

router or host that is incapable of supporting the requested reservation may send an error to the receiver, which is then free to perhaps try a lesser reservation.

The z/OS UNIX RSVP agent can provide actual resource reservations on ATM interfaces. The RSVP agent passes the reservation request to the TCP/IP stack, where a bandwidth reserved SVC is established on the ATM link to support the reservation request. The RSVP agent can also cause the Type of Service (TOS) byte to be set for any given RSVP flow, by using the OutgoingTOS keyword on the PolicyAction statement of a defined service policy.

Gathering diagnostic information

The RSVP Agent writes logging information to a log file. The level of logged information is controlled by the LogLevel configuration statement. By default, only error and warning messages are written. To gather more diagnostic information, you can specify a LogLevel value. The maximum information is logged with a LogLevel value of 511. Refer to the *z/OS Communications Server: IP Configuration Guide* for more details on using LogLevel, as well as the location of the log file.

Certain other information can be useful in diagnosing RSVP Agent problems:

- Output from the TSO NETSTAT SLAP or onetstat -j commands
- Output from the pasearch command for RSVP scoped policies
- SNMP output from walks of the SLA Subagent MIB tables
- TCP/IP CTRACE output, using the INTERNET and IOCTL CTRACE options
- Policy Agent log output if RSVP scoped policies are defined

Diagnosing RSVP agent problems

Problems with the RSVP agent generally fall into one of the following categories:

- Initialization Problems
- Application Problems
- Service Policy Problems

Initialization problems

If the RSVP Agent does not complete initialization, run it with LogLevel set to 511 and check the log file for error conditions. Common problems could include:

- RSVP Agent not authorized to security product.

The RSVP Agent must be authorized to a security product profile. Refer to the *z/OS Communications Server: IP Configuration Guide* for details on setting up the proper authorization.

- Unable to read configuration file.

Is the correct configuration file specified? Refer to the *z/OS Communications Server: IP Configuration Guide* for the search order used to locate the configuration file. Does the file exist? Are the permission bits correctly set for an HFS file?

- Unable to associate with the TCP/IP stack.

Is the associated TCP/IP stack started? The RSVP Agent uses the TCP/IP image name specified in the configuration file, or uses the standard resolver search order, to locate the name of the TCP/IP stack. The log file indicates the stack name being used.

- Unable to initialize interfaces.

The RSVP Agent needs to initialize each interface for which it is configured. A pair of "mailboxes" are created for each interface. Check for error messages while creating the "rsvp" and "rsvp-udp" mailboxes for each interface. An error received while trying to join a multicast group on an interface that is not multicast capable is expected, and looks like:

```
WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed -  
EDC5121I Invalid argument.
```

Application problems

If a Qos-aware application using RAPI is experiencing problems, check the following:

- RAPI DLL not found.

An application using RAPI must have access to the RAPI DLL at run time. This is normally accomplished with the LIBPATH environment variable. Check that the LIBPATH environment variable is specified and that it contains the directory in which the RAPI DLL (rapi.dll) resides, which should be /usr/lib.

- Error RAPI_ERR_NORSVP received.

If the application receives a RAPI_ERR_NORSVP error code when calling a RAPI function, ensure that the RSVP Agent has been successfully started.

Policy problems

Policies with RSVP scope can be defined and made available by way of the Policy Agent. If problems are encountered using such policies, check the following:

- RSVP policies not being applied to data flows.

If the limits imposed by defined RSVP-scoped policies are not taking effect, check that the Policy Agent has been successfully started. The Policy Agent must be active in order for the RSVP Agent to retrieve these policies. Check that the policies are correctly defined. For example, do not specify both inbound and outbound interfaces in a single policy condition because such a policy will never map to any traffic on an end host node. Also, check both the RSVP Agent and Policy Agent log files for errors dealing with obtaining policies.

- Policy values not being used or are incorrect.

If the values being used in the policies to limit Tspec and Rspec values do not appear to be correct, or do not seem to be applied to RSVP data traffic, be aware that the service policy and Tspec/Rspec units of measure are different. Specifically, the following are different:

Service Policy Unit	Tspec/Rspec Unit
MaxRatePerFlow: kilobits/second	r/R: bytes/second
MaxTokenBucketPerFlow: kilobits	b: bytes

To arrive at the values to specify on the service policy, multiply the target Tspec/Rspec value by 8, then divide by 1000. For example, if the target Tspec b value is 6000, the corresponding MaxTokenBucketPerFlow value is 48 ($6000 \times 8 / 1000 = 48$). See Chapter 21, "Diagnosing Policy Agent problems" on page 531 for more information about Policy Agent.

Example log file

Figure 64 on page 566 demonstrates some of the RSVP Agent processing. This log file was created using a LogLevel of 511.

Lines with numbers displayed like **1** are annotations that are described following the log.

```
01
03/22 08:51:01 INFO  :.main: ***** RSVP Agent started *****
02
03/22 08:51:01 INFO  :...locate_configFile: Specified configuration file: /u/user10/rsvpd1.conf
03/22 08:51:01 INFO  :.main: Using log level 511
03/22 08:51:01 INFO  :..settcpimage: Get TCP images rc - EDC8112I Operation not supported on socket.
03
03/22 08:51:01 INFO  :..settcpimage: Associate with TCP/IP image name = TCPCS
03/22 08:51:02 INFO  :..reg_process: registering process with the system
03/22 08:51:02 INFO  :..reg_process: attempt 05/390 registration
03/22 08:51:02 INFO  :..reg_process: return from registration rc=0
04
03/22 08:51:06 TRACE  :...read_physical_netif: Home list entries returned = 7
03/22 08:51:06 INFO  :...read_physical_netif: index #0, interface VLINK1 has address 129.1.1.1, ifidx 0
03/22 08:51:06 INFO  :...read_physical_netif: index #1, interface TR1 has address 9.37.65.139, ifidx 1
03/22 08:51:06 INFO  :...read_physical_netif: index #2, interface LINK11 has address 9.67.100.1, ifidx 2
03/22 08:51:06 INFO  :...read_physical_netif: index #3, interface LINK12 has address 9.67.101.1, ifidx 3
03/22 08:51:06 INFO  :...read_physical_netif: index #4, interface CTCD0 has address 9.67.116.98, ifidx 4
03/22 08:51:06 INFO  :...read_physical_netif: index #5, interface CTCD2 has address 9.67.117.98, ifidx 5
03/22 08:51:06 INFO  :...read_physical_netif: index #6, interface LOOPBACK has address 127.0.0.1, ifidx 0
03/22 08:51:06 INFO  :...mailslot_create: creating mailslot for timer
03/22 08:51:06 INFO  :...mailbox_register: mailbox allocated for timer
05
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP via UDP
06
03/22 08:51:06 WARNING:....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address not available.
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 129.1.1.1, entity for rsvp allocated and initialized
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 WARNING:....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address not available.
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 INFO  :..entity_initialize: interface 9.67.100.1, entity for rsvp allocated and initialized
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 WARNING:....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address not available.
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.67.101.1, entity for rsvp allocated and initialized
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.67.116.98, entity for rsvp allocated and initialized
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO  :....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 INFO  :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.67.117.98, entity for rsvp allocated and initialized
```

Figure 64. RSVP Agent processing log (Part 1 of 6)


```

03/22 08:51:06 INFO :.....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO :.....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 INFO :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE :..entity_initialize: interface 127.0.0.1, entity for rsvp allocated and initialized
03/22 08:51:06 INFO :.....mailslot_create: creating socket for querying route
03/22 08:51:06 INFO :.....mailbox_register: no mailbox necessary for forward
03/22 08:51:06 INFO :.....mailslot_create: creating mailslot for route engine - informational socket
03/22 08:51:06 TRACE :.....mailslot_create: ready to accept informational socket connection
03/22 08:51:11 INFO :.....mailbox_register: mailbox allocated for route
03/22 08:51:11 INFO :.....mailslot_create: creating socket for traffic control module
03/22 08:51:11 INFO :....mailbox_register: no mailbox necessary for traffic-control
03/22 08:51:11 INFO :.....mailslot_create: creating mailslot for RSVP client API
03/22 08:51:11 INFO :...mailbox_register: mailbox allocated for rsvp-api
03/22 08:51:11 INFO :...mailslot_create: creating mailslot for terminate
03/22 08:51:11 INFO :..mailbox_register: mailbox allocated for terminate
03/22 08:51:11 INFO :...mailslot_create: creating mailslot for dump
03/22 08:51:11 INFO :..mailbox_register: mailbox allocated for dump
03/22 08:51:11 INFO :...mailslot_create: creating mailslot for (broken) pipe
03/22 08:51:11 INFO :..mailbox_register: mailbox allocated for pipe
07
03/22 08:51:11 INFO :.main: rsvpd initialization complete
08
03/22 08:52:50 INFO :.....rsvp_api_open: accepted a new connection for rapi
03/22 08:52:50 INFO :.....mailbox_register: mailbox allocated for mailbox
03/22 08:52:50 TRACE :.....rsvp_event_mapSession: Session=9.67.116.99:1047:6 does not exist
09
03/22 08:52:50 EVENT :.....api_reader: api request SESSION
10
03/22 08:52:50 TRACE :.....rsvp_event_establishSession: local node will send
03/22 08:52:50 INFO :.....router_forward_getOI: Ioctl to get route entry successful
03/22 08:52:50 TRACE :.....router_forward_getOI: source address: 9.67.116.98
03/22 08:52:50 TRACE :.....router_forward_getOI: out inf: 9.67.116.98
03/22 08:52:50 TRACE :.....router_forward_getOI: gateway: 0.0.0.0
03/22 08:52:50 TRACE :.....router_forward_getOI: route handle: 7f5251c8
11
03/22 08:52:50 TRACE :.....event_establishSessionSend: found outgoing if=9.67.116.98 through
forward engine
03/22 08:52:50 TRACE :.....rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
12
03/22 08:52:50 EVENT :.....api_reader: api request SENDER
13
03/22 08:52:50 INFO :.....init_policyAPI: papi_debug: Entering
03/22 08:52:50 INFO :.....init_policyAPI: papi_debug: papiLogFunc = 98681F0 papiUserValue = 0
03/22 08:52:50 INFO :.....init_policyAPI: papi_debug: Exiting
03/22 08:52:50 INFO :.....init_policyAPI: APIInitialize: Entering
03/22 08:52:50 INFO :.....init_policyAPI: open_socket: Entering
03/22 08:52:50 INFO :.....init_policyAPI: open_socket: Exiting
03/22 08:52:50 INFO :.....init_policyAPI: APIInitialize: ApiHandle = 98BDFB0, connfd = 22
03/22 08:52:50 INFO :.....init_policyAPI: APIInitialize: Exiting
03/22 08:52:50 INFO :.....init_policyAPI: RegisterWithPolicyAPI: Entering

```

Figure 64. RSVP Agent processing log (Part 2 of 6)

```

03/22 08:52:50 INFO :.....init_policyAPI: RegisterWithPolicyAPI: Writing to socket = 22
03/22 08:52:50 INFO :.....init_policyAPI: ReadBuffer: Entering
03/22 08:52:51 INFO :.....init_policyAPI: ReadBuffer: Exiting
03/22 08:52:51 INFO :.....init_policyAPI: RegisterWithPolicyAPI: Exiting
03/22 08:52:51 INFO :.....init_policyAPI: Policy API initialized
03/22 08:52:51 INFO :.....rpapi_getPolicyData: RSVPFindActionName: Entering
03/22 08:52:51 INFO :.....rpapi_getPolicyData: ReadBuffer: Entering
03/22 08:52:51 INFO :.....rpapi_getPolicyData: ReadBuffer: Exiting
03/22 08:52:51 INFO :.....rpapi_getPolicyData: RSVPFindActionName: Result = 0
03/22 08:52:51 INFO :.....rpapi_getPolicyData: RSVPFindActionName: Exiting

14
03/22 08:52:51 INFO :.....rpapi_getPolicyData: found action name CLCat2 for
flow[sess=9.67.116.99:1047:6,source=9.67.116.98:8000]
03/22 08:52:51 INFO :.....rpapi_getPolicyData: RSVPFindServiceDetailsOnActName: Entering
03/22 08:52:51 INFO :.....rpapi_getPolicyData: ReadBuffer: Entering
03/22 08:52:51 INFO :.....rpapi_getPolicyData: ReadBuffer: Exiting
03/22 08:52:51 INFO :.....rpapi_getPolicyData: RSVPFindServiceDetailsOnActName: Result = 0
03/22 08:52:51 INFO :.....rpapi_getPolicyData: RSVPFindServiceDetailsOnActName: Exiting
03/22 08:52:51 INFO :.....api_reader: appl chose service type 1
03/22 08:52:51 INFO :.....rpapi_getSpecData: RSVPGetTSpec: Entering
03/22 08:52:51 INFO :.....rpapi_getSpecData: RSVPGetTSpec: Result = 0
03/22 08:52:51 INFO :.....rpapi_getSpecData: RSVPGetTSpec: Exiting
03/22 08:52:51 TRACE :.....api_reader: new service=1, old service=0
03/22 08:52:51 INFO :.....rsvp_flow_stateMachine: state SESSIONED, event PATHDELTA

15
03/22 08:52:51 TRACE :.....rsvp_action_nHop: constructing a PATH
03/22 08:52:51 TRACE :.....flow_timer_start: started T1

16
03/22 08:52:51 TRACE :.....rsvp_flow_stateMachine: entering state PATHED
03/22 08:52:51 TRACE :.....mailslot_send: sending to (9.67.116.99:0)
03/22 08:52:51 TRACE :.....mailslot_send: sending to (9.67.116.99:1698)

17
03/22 08:52:51 TRACE :.....rsvp_event: received event from RAW-IP on interface 9.67.116.98
03/22 08:52:51 TRACE :.....rsvp_explode_packet: v=1,flg=0,type=2,cksm=54875,ttl=255,rsv=0,len=84
03/22 08:52:51 TRACE :.....rsvp_parse_objects: STYLE is WF
03/22 08:52:51 INFO :.....rsvp_parse_objects: obj RSVP_HOP hop=9.67.116.99, lih=0
03/22 08:52:51 TRACE :.....rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:52:51 INFO :.....rsvp_flow_stateMachine: state PATHED, event RESVDELTA

18
03/22 08:52:51 TRACE :.....traffic_action_oif: is to install filter
03/22 08:52:51 INFO :.....qosmgr_request: src-9.67.116.98:8000 dst-9.67.116.99:1047 proto-6
rthdl-7f5251c8

19
03/22 08:52:51 INFO :.....qosmgr_request: [CL r=90000 b=6000 p=110000 m=1024 M=2048]
03/22 08:52:51 INFO :.....qosmgr_request: Ioctl to add reservation successful
03/22 08:52:51 INFO :.....rpapi_Reg_UnregFlow: RSVPPutActionName: Entering

```

Figure 64. RSVP Agent processing log (Part 3 of 6)

```

03/22 08:52:51 INFO :.....rpapi_Reg_UnregFlow: ReadBuffer: Entering
03/22 08:52:52 INFO :.....rpapi_Reg_UnregFlow: ReadBuffer: Exiting
03/22 08:52:52 INFO :.....rpapi_Reg_UnregFlow: RSVPPutActionName: Result = 0
03/22 08:52:52 INFO :.....rpapi_Reg_UnregFlow: RSVPPutActionName: Exiting
03/22 08:52:52 INFO :.....rpapi_Reg_UnregFlow: flow[sess=9.67.116.99:1047:6,
source=9.67.116.98:8000] registered with CLCat2
03/22 08:52:52 EVENT :.....qosmgr_response: RESVRESP from qosmgr, reason=0, qoshandle=8b671d0
03/22 08:52:52 INFO :.....qosmgr_response: src=9.67.116.98:8000 dst=9.67.116.99:1047 proto=6
03/22 08:52:52 TRACE :.....traffic_reader: tc response msg=1, status=1
03/22 08:52:52 INFO :.....traffic_reader: Reservation req successful[session=9.67.116.99:1047:6,
source=9.67.116.98:8000, qoshd=8b671d0]
20
03/22 08:52:52 TRACE :.....api_action_sender: constructing a RESV
03/22 08:52:52 TRACE :.....flow_timer_stop: stopped T1
03/22 08:52:52 TRACE :.....flow_timer_stop: Stop T4
03/22 08:52:52 TRACE :.....flow_timer_start: started T1
03/22 08:52:52 TRACE :.....flow_timer_start: Start T4
21
03/22 08:52:52 TRACE :.....rsvp_flow_stateMachine: entering state RESVED
22
03/22 08:53:07 EVENT :..mailslot_sitter: process received signal SIGALRM
03/22 08:53:07 TRACE :.....event_timerT1_expire: T1 expired
03/22 08:53:07 INFO :.....router_forward_getOI: Ioctl to query route entry successful
03/22 08:53:07 TRACE :.....router_forward_getOI: source address: 9.67.116.98
03/22 08:53:07 TRACE :.....router_forward_getOI: out inf: 9.67.116.98
03/22 08:53:07 TRACE :.....router_forward_getOI: gateway: 0.0.0.0
03/22 08:53:07 TRACE :.....router_forward_getOI: route handle: 7f5251c8
03/22 08:53:07 INFO :.....rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:53:07 TRACE :.....rsvp_action_nHop: constructing a PATH
03/22 08:53:07 TRACE :.....flow_timer_start: started T1
03/22 08:53:07 TRACE :.....rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:07 TRACE :.....mailslot_send: sending to (9.67.116.99:0)
23
03/22 08:53:22 TRACE :.....rsvp_event: received event from RAW-IP on interface 9.67.116.98
03/22 08:53:22 TRACE :.....rsvp_explode_packet: v=1,flg=0,type=2,cksm=54875,ttl=255,rsv=0,len=84
03/22 08:53:22 TRACE :.....rsvp_parse_objects: STYLE is WF
03/22 08:53:22 INFO :.....rsvp_parse_objects: obj RSVP_HOP hop=9.67.116.99, lih=0
03/22 08:53:22 TRACE :.....rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:53:22 INFO :.....rsvp_flow_stateMachine: state RESVED, event RESV
03/22 08:53:22 TRACE :.....flow_timer_stop: Stop T4
03/22 08:53:22 TRACE :.....flow_timer_start: Start T4
03/22 08:53:22 TRACE :.....rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:22 EVENT :..mailslot_sitter: process received signal SIGALRM
03/22 08:53:22 TRACE :.....event_timerT1_expire: T1 expired
03/22 08:53:22 INFO :.....router_forward_getOI: Ioctl to query route entry successful
03/22 08:53:22 TRACE :.....router_forward_getOI: source address: 9.67.116.98
03/22 08:53:22 TRACE :.....router_forward_getOI: out inf: 9.67.116.98
03/22 08:53:22 TRACE :.....router_forward_getOI: gateway: 0.0.0.0
03/22 08:53:22 TRACE :.....router_forward_getOI: route handle: 7f5251c8
03/22 08:53:22 INFO :.....rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:53:22 TRACE :.....rsvp_action_nHop: constructing a PATH
03/22 08:53:22 TRACE :.....flow_timer_start: started T1
03/22 08:53:22 TRACE :.....rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:22 TRACE :.....mailslot_send: sending to (9.67.116.99:0)
03/22 08:53:38 EVENT :..mailslot_sitter: process received signal SIGALRM
03/22 08:53:38 TRACE :.....event_timerT1_expire: T1 expired
03/22 08:53:38 INFO :.....router_forward_getOI: Ioctl to query route entry successful

```

Figure 64. RSVP Agent processing log (Part 4 of 6)

```

03/22 08:53:38 TRACE :.....router_forward_getOI:      source address:  9.67.116.98
03/22 08:53:38 TRACE :.....router_forward_getOI:      out inf:   9.67.116.98
03/22 08:53:38 TRACE :.....router_forward_getOI:      gateway:   0.0.0.0
03/22 08:53:38 TRACE :.....router_forward_getOI:      route handle:  7f5251c8
03/22 08:53:38 INFO  :.....rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:53:38 TRACE :.....rsvp_action_nHop: constructing a PATH
03/22 08:53:38 TRACE :.....flow_timer_start: started T1
03/22 08:53:38 TRACE :.....rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:38 TRACE :.....mailslot_send: sending to (9.67.116.99:0)
03/22 08:53:52 TRACE :.....rsvp_event: received event from RAW-IP on interface 9.67.116.98
03/22 08:53:52 TRACE :.....rsvp_explode_packet: v=1,flg=0,type=2,cksm=54875,ttl=255,rsv=0,len=84
03/22 08:53:52 TRACE :.....rsvp_parse_objects: STYLE is WF
03/22 08:53:52 INFO  :.....rsvp_parse_objects: obj RSVP_HOP hop=9.67.116.99, lih=0
03/22 08:53:52 TRACE :.....rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:53:52 INFO  :.....rsvp_flow_stateMachine: state RESVED, event RESV
03/22 08:53:52 TRACE :.....flow_timer_stop: Stop T4
03/22 08:53:52 TRACE :.....flow_timer_start: Start T4
03/22 08:53:52 TRACE :.....rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:53 EVENT :..mailslot_sitter: process received signal SIGALRM
03/22 08:53:53 TRACE :.....event_timerT1_expire: T1 expired
03/22 08:53:53 INFO  :.....router_forward_getOI: Ioctl to query route entry successful
03/22 08:53:53 TRACE :.....router_forward_getOI:      source address:  9.67.116.98
03/22 08:53:53 TRACE :.....router_forward_getOI:      out inf:   9.67.116.98
03/22 08:53:53 TRACE :.....router_forward_getOI:      gateway:   0.0.0.0
03/22 08:53:53 TRACE :.....router_forward_getOI:      route handle:  7f5251c8
03/22 08:53:53 INFO  :.....rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:53:53 TRACE :.....rsvp_action_nHop: constructing a PATH
03/22 08:53:53 TRACE :.....flow_timer_start: started T1
03/22 08:53:53 TRACE :.....rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:53 TRACE :.....mailslot_send: sending to (9.67.116.99:0)
03/22 08:54:09 EVENT :..mailslot_sitter: process received signal SIGALRM
03/22 08:54:09 TRACE :.....event_timerT1_expire: T1 expired
03/22 08:54:09 INFO  :.....router_forward_getOI: Ioctl to query route entry successful
03/22 08:54:09 TRACE :.....router_forward_getOI:      source address:  9.67.116.98
03/22 08:54:09 TRACE :.....router_forward_getOI:      out inf:   9.67.116.98
03/22 08:54:09 TRACE :.....router_forward_getOI:      gateway:   0.0.0.0
03/22 08:54:09 TRACE :.....router_forward_getOI:      route handle:  7f5251c8
03/22 08:54:09 INFO  :.....rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:54:09 TRACE :.....rsvp_action_nHop: constructing a PATH
03/22 08:54:09 TRACE :.....flow_timer_start: started T1
03/22 08:54:09 TRACE :.....rsvp_flow_stateMachine: reentering state RESVED
03/22 08:54:09 TRACE :.....mailslot_send: sending to (9.67.116.99:0)
03/22 08:54:22 TRACE :.....rsvp_event: received event from RAW-IP on interface 9.67.116.98
03/22 08:54:22 TRACE :.....rsvp_explode_packet: v=1,flg=0,type=2,cksm=54875,ttl=255,rsv=0,len=84
03/22 08:54:22 TRACE :.....rsvp_parse_objects: STYLE is WF
03/22 08:54:22 INFO  :.....rsvp_parse_objects: obj RSVP_HOP hop=9.67.116.99, lih=0
03/22 08:54:22 TRACE :.....rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:54:22 INFO  :.....rsvp_flow_stateMachine: state RESVED, event RESV
03/22 08:54:22 TRACE :.....flow_timer_stop: Stop T4
03/22 08:54:22 TRACE :.....flow_timer_start: Start T4
03/22 08:54:22 TRACE :.....rsvp_flow_stateMachine: reentering state RESVED
03/22 08:54:24 EVENT :..mailslot_sitter: process received signal SIGALRM
03/22 08:54:24 TRACE :.....event_timerT1_expire: T1 expired
03/22 08:54:24 INFO  :.....router_forward_getOI: Ioctl to query route entry successful
03/22 08:54:24 TRACE :.....router_forward_getOI:      source address:  9.67.116.98
03/22 08:54:24 TRACE :.....router_forward_getOI:      out inf:   9.67.116.98
03/22 08:54:24 TRACE :.....router_forward_getOI:      gateway:   0.0.0.0
03/22 08:54:24 TRACE :.....router_forward_getOI:      route handle:  7f5251c8

```

Figure 64. RSVP Agent processing log (Part 5 of 6)

```

03/22 08:54:24 INFO :.....rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:54:24 TRACE :.....rsvp_action_nHop: constructing a PATH
03/22 08:54:24 TRACE :.....flow_timer_start: started T1
03/22 08:54:24 TRACE :.....rsvp_flow_stateMachine: reentering state RESVED
03/22 08:54:24 TRACE :.....mailslot_send: sending to (9.67.116.99:0)
03/22 08:54:35 TRACE :.....rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
24
03/22 08:54:35 EVENT :.....api_reader: api request SENDER_WITHDRAW
03/22 08:54:35 INFO :.....rsvp_flow_stateMachine: state RESVED, event PATHTEAR
25
03/22 08:54:35 TRACE :.....traffic_action_oif: is to remove filter
03/22 08:54:35 INFO :.....qosmgr_request: Ioctl to remove reservation successful
03/22 08:54:35 INFO :.....rpapi_Reg_UnregFlow: RSVPRemActionName: Entering

03/22 08:54:35 INFO :.....rpapi_Reg_UnregFlow: ReadBuffer: Entering

03/22 08:54:35 INFO :.....rpapi_Reg_UnregFlow: ReadBuffer: Exiting

03/22 08:54:35 INFO :.....rpapi_Reg_UnregFlow: RSVPRemActionName: Result = 0

03/22 08:54:35 INFO :.....rpapi_Reg_UnregFlow: RSVPRemActionName: Exiting

03/22 08:54:35 INFO :.....rpapi_Reg_UnregFlow: flow[sess=9.67.116.99:1047:6,
source=9.67.116.98:8000] unregistered from CLCat2
03/22 08:54:35 EVENT :.....qosmgr_response: DELRESP from qosmgr, reason=0, qoshandle=0
03/22 08:54:35 INFO :.....qosmgr_response: src=9.67.116.98:8000 dst=9.67.116.99:1047 proto=6
03/22 08:54:35 TRACE :.....traffic_reader: tc response msg=3, status=1
26
03/22 08:54:35 TRACE :.....rsvp_action_nHop: constructing a PATHTEAR
03/22 08:54:35 TRACE :.....flow_timer_stop: stopped T1
03/22 08:54:35 TRACE :.....flow_timer_stop: Stop T4
27
03/22 08:54:35 TRACE :.....rsvp_flow_stateMachine: entering state SESSIONED
03/22 08:54:35 TRACE :.....mailslot_send: sending to (9.67.116.99:0)
03/22 08:54:35 TRACE :.....rsvp_event_propagate: flow[session=9.67.116.99:1047:6,
source=9.67.116.98:8000] ceased
28
03/22 08:54:35 EVENT :.....api_reader: api request CLOSE
03/22 08:54:35 INFO :.....rsvp_flow_stateMachine: state SESSIONED, event PATHTEAR
03/22 08:54:35 PROTERR:.....rsvp_flow_stateMachine: state SESSIONED does not expect event PATHTEAR
29
03/22 08:54:53 EVENT :..mailslot_sitter: process received signal SIGTERM
03/22 08:54:53 INFO :...check_signals: received TERM signal
03/22 08:54:53 INFO :.....term_policyAPI: UnRegisterFromPolicyAPI: Entering

03/22 08:54:53 INFO :.....term_policyAPI: ReadBuffer: Entering

03/22 08:54:53 INFO :.....term_policyAPI: ReadBuffer: Exiting

03/22 08:54:53 INFO :.....term_policyAPI: UnRegisterFromPolicyAPI: Result = 0

03/22 08:54:53 INFO :.....term_policyAPI: UnRegisterFromPolicyAPI: Exiting

03/22 08:54:53 INFO :.....term_policyAPI: APITerminate: Entering

03/22 08:54:53 INFO :.....term_policyAPI: APITerminate: Exiting

03/22 08:54:53 INFO :.....term_policyAPI: Policy API terminated
03/22 08:54:53 INFO :.....dreg_process: deregistering process with the system
03/22 08:54:53 INFO :.....dreg_process: attempt to dereg (ifaeddrb_byaddr)
03/22 08:54:53 INFO :.....dreg_process: rc from ifaeddrb_byaddr rc =0
03/22 08:54:53 INFO :.....terminator: process terminated with exit code 0

```

Figure 64. RSVP Agent processing log (Part 6 of 6)

Following are short descriptions of the numbered items in the trace:

- 01** The RSVP Agent is started.
- 02** The configuration file being used is reported.
- 03** The name of the TCP/IP stack that the RSVP Agent associates itself with is reported.
- 04** The name and IP address of the interfaces configured to the associated stack are reported. Note that the RSVP Agent gets notified by the stack of any interface additions, deletions, or changes after this point.
- 05** The interfaces are initialized one by one.
- 06** Some interface types are not enabled for multicasting. Therefore, when the RSVP Agent tries to enable multicasting, a warning is reported. Such interfaces can still be used for unicasting.
- 07** RSVP Agent initialization is complete.
- 08** An application makes its first RAPI call, initializing the RAPI interface with the RSVP Agent.
- 09** The type of RAPI request is SESSION, meaning a `rapi_session()` call was made.
- 10** The RSVP Agent determines that the application will send based on the specified destination address not being a local interface.
- 11** The outbound interface to use for the session is returned from the stack.
- 12** The application issues a `rapi_sender()` call, passing the `Tspec`.
- 13** The Policy Agent interface is initialized.
- 14** The policy action "CLCat2" is obtained from the Policy Agent for the specified flow.
- 15** The RSVP Agent constructs an RSVP PATH packet to be sent to the destination.
- 16** The flow enters the pathed stated (PATHED), meaning a PATH packet has been sent for the flow.
- 17** An RSVP RESV packet is received from the RSVP Agent at the receiver node, specifying the reservation parameters.
- 18** The RSVP Agent installs the reservation request into the TCP/IP stack and registers the flow with the Policy Agent.
- 19** The type of reservation request is shown (CL, for Controlled Load) along with the reservation parameters (the `r`, `b`, `p`, `m`, `M` values in `Tspec` format).
- 20** The RESV packet values are passed to the sender application.
- 21** The flow enters the reserved state (RESVED), meaning the reservation has been put in place and the RESV packet has been forwarded to the previous hop (in this case the sender application).
- 22** A T1 timeout occurs, meaning a PATH refresh packet is sent. This occurs every 15 seconds.
- 23** A refreshed RESV packet is received from the RSVP Agent at the receiver node. This occurs every 30 seconds.
- 24** The application issues a `rapi_release()` call to end the RAPI session.
- 25** The reservation is removed from the TCP/IP stack and unregistered from the Policy Agent.

- 26** A PATHTEAR packet is constructed and sent, to tear down the flow along the data path.
- 27** The flow enters the sessioned state (SESSIONED), meaning that the flow has been torn down.
- 28** The application closes the API session, resulting in an error being reported because the state of the flow is SESSIONED. This error can be ignored.
- 29** A SIGTERM signal is received (due to a kill command issued from the UNIX shell), and the RSVP Agent shuts itself down.

Chapter 23. Diagnosing intrusion detection problems

This chapter provides information and guidance to diagnose Intrusion Detection Service (IDS) problems, including traffic regulation management daemon (TRMD) related problems.

Overview

The Intrusion Detection Services policy is installed into the stack by the Policy Agent (PAGENT). Once the policy is installed, IDS detects, processes, and reports on events as requested by the policy. TRMD, part of IDS, handles reporting IDS statistics and events to syslogd. Areas where problems may occur are:

- Policy installation
- Output to syslogd, the console, or the IDS trace missing or volume too high
- TRMD initialization

Diagnosing IDS policy problems

Use `pasearch -i` (refer to *z/OS Communications Server: IP System Administrator's Commands*) to see what IDS policies are active in Policy Agent. If you are running multiple stacks, ensure that `pasearch` is reporting on the stack you are interested in. See Chapter 21, "Diagnosing Policy Agent problems" on page 531 if you do not see the IDS policies expected.

Use `NETSTAT IDS` or `onetstat -k` (refer to *z/OS Communications Server: IP System Administrator's Commands*) to see how your IDS policies have been mapped by the stack. If you are running multiple stacks, ensure that your resolver configuration correctly identifies the stack you are interested in. Ensure that your IDS policies are correctly defined. Refer to IDS policy considerations in the *z/OS Communications Server: IP Configuration Guide* and the section about the IDS attribute matrix in the *z/OS Communications Server: IP Configuration Reference*.

Note that some IDS policies are not mapped until they are needed. Attack, Scan_Global and Scan_Event for protocol ICMP are mapped immediately when the policy is installed in the stack. Scan_Event policies for protocols TCP and UDP are mapped on the first occurrence of a potentially countable event. TR policies for protocol TCP are mapped when a local application does a `listen()` and when a client completes the three-way connection handshake. TR policies for protocol UDP are mapped when an inbound datagram arrives for a bound port.

Diagnosing IDS output problems

The following describe diagnostic steps for some problems you may encounter.

IDS SyslogD output is missing

Perform the following to determine the cause for IDS syslogd missing.

1. Ensure that Policy Agent is running on this system.
2. Ensure that TRMD is running for this stack on this system. Consider using TCPIP PROFILE AutoLog for TRMD.
3. Ensure that syslogd is running on this system.
4. Ensure that syslogd is configured for IDS output:
 - TRMD always writes to the syslog daemon facility.

- Events are written to the syslog level configured in the relevant policy. Statistics are always written to INFO level.
- If running multiple TRMDs, consider using trmd jobname prefix to separate IDS output by stack.

IDS console output

Under certain conditions, IDS suppresses console messages to avoid flooding the system console.

Scan detection is reported at most once per fast scan interval for a particular source IP address. If a scan is continually detected for the same source IP address, consider adding this address to your scan exclusion list (if this user is legitimately accessing resources). The installation also has the option of requesting notification to syslogd rather than to the console. The same criteria is used for reporting scans to syslogd as to the console.

IDS attack policy actions support the **idsMaxEventMessage** attribute. If specified, this limits the number of times the same **idsAttackType** will be reported to the system console within any 5-minute time period.

Traffic regulation for protocol TCP suppresses console reporting of three events that could occur repeatedly.

- Only the first connection denied, when an application exceeds the **idsTRtcpTotalConnections** limit, is reported during each port constrained period.
- Only the first connection denied, when a source host exceeds the **idsTRtcpPercentage** available limit, is reported until the number of connections by that source host to this application drops below 88% of the limit and at least 2 connections below the limit.
- Connections that would exceed the **idsTRtcpPercentage** of available connections per source host, but are allowed because of a higher value in QoS policy, are reported to syslogd only.

IDS packet trace output

- See “Intrusion Detection Services trace (SYSTCPIS)” on page 136 if message **EZZ4210I CTRACE DEFINE FAILED FOR CTIIDS00** is issued at stack initialization.
- Consider starting the MVS external writer. See “Formatting packet traces using IPCS” on page 95 for information on formatting the IDS packet trace in a dump.
- For IDS attack policy, the tracing action allows packets associated with attack events to be traced. For all attack categories except flood, a single packet triggers an event and the packet is traced. To prevent trace flooding, a maximum of 100 attack packets per attack category are traced within a 5-minute interval. For the flood category, the first 100 packets discarded during the flood are traced.

Unusual conditions

Most messages issued by IDS relate to the detection of an IDS condition. However, the messages mentioned below should be investigated because they signal conditions which affect IDS normal processing that may result in IDS information being lost or delayed.

Buffer overflow transferring message data between the stack and TRMD

The following messages in syslog indicate that IDS events or statistics are being generated at a rate that is overflowing internal buffers used to relay the messages from the stack to TRMD. These messages are a warning that actual event or statistics messages are missing from the syslog. If these messages occur frequently, then IDS policy changes are necessary to reduce the amount of IDS logging, or the amount of statistics information, being generated.

```
EZZ9325I TRMD Log records missing: logtype,logmissing  
EZZ9326I TRMD Statistics records missing: stattype,statmissing
```

Repeated attacks of the same type at a high rate

A message is issued to indicate that attack policy is in place and the attack type indicated is occurring repeatedly at a high rate. To avoid flooding syslog and conserve system resources, a maximum of 100 event messages per attack type are logged to syslogd within a 5-minute interval. This limit is always in effect. The following message indicates the number of duplicate attacks for which messages have been suppressed.

```
EZZ9327I TRMD Attack log records suppressed: attack_type,count
```

Scan storage constrained

The following is an example of a message issued if scan detection attempted to obtain storage in order to track a potential scan event and could not obtain the required amount of storage.

```
EZZ8761I IDS EVENT DETECTED  
EZZ8762I EVENT TYPE: SCAN STORAGE CONSTRAINED  
EZZ8763I CORRELATOR 0 - PROBEID 0300FFF3  
EZZ8766I IDS RULE N/A  
EZZ8767I IDS ACTION N/A
```

Processing will continue without adding the tracking information for this packet or for subsequent packets in the current internal interval (an internal interval is either 30 or 60 seconds). This could result in missing potential scan events.

The installation should attempt to determine the cause of the storage shortage. Scan detection itself can potentially consume large amounts of storage and should be looked at as part of the problem determination. There are two ways to determine whether scan is consuming large amounts of storage.

- Message EZZ8768I (EZZ8768I IDS SCAN STORAGE EXCEEDED *nbrmeg* MB, TRACKING *nbrsip* SOURCE IP ADDRESSES) is issued after scan detection acquires more than a megabyte of storage. This message is reissued at each power of 2 MB increments (for example, 1 MB, 2 MB, 4 MB, 8 MB, and so forth).
- The Netstat IDS command displays high level scan information. For example:

```
SCAN DETECTION:  
  GLOBRULENAME: IDS-RULE4  
  ICMRULENAME: IDS-RULE8  
  TOTDETECTED: 1          DETCURRPLC: 1  
  DETCURRINT: 0          INTERVAL: 30  
  SRCIPSTRKD: 125        STRGLEV: 00000M
```

The SRCIPSTRKD field indicates the number of source IPs being tracked and the STRGLEV field indicates the number of megabytes of storage that scan is holding.

If scan processing is contributing to the storage shortage, consider changing the scan policy. If the installation has set the scan sensitivity to HIGH on high usage ports, consider reducing the sensitivity level or removing the port from scan detection until the storage constraint is resolved.

When scan starts to successfully obtain storage again, a SCAN STORAGE UNCONSTRAINED message will be issued.

Excessive processing time for scans

The following is an example of a message issued as a result of excessive processing time for scans.

```
EZZ8761I IDS EVENT DETECTED
EZZ8762I EVENT TYPE: SCAN INTERVAL OVERRUN
EZZ8763I CORRELATOR 0 - PROBEID 0300FFF5
EZZ8766I IDS RULE N/A
EZZ8767I IDS ACTION N/A
```

If an installation repeatedly receives this message, scan processing is not able to complete its evaluation of the source IP addresses it is tracking in its normal interval (either 30 or 60 seconds). This could delay the detection of subsequent scans. This most likely indicates that a large number of source IP addresses are being monitored. If the policy is using high scan sensitivity, the installation should consider lowering the scan sensitivity level for high usage ports.

Diagnosing TRMD problems

The most common type of TRMD problem is initialization.

The TRMD writes logging information to a log file. The level of logged information is controlled by the -d startup option. To gather more diagnostic information, you can start the TRMD with the -d startup option. The maximum information is logged with the -d 3 option. Log output is directed to the syslog daemon (syslogd). Refer to the *z/OS Communications Server: IP Configuration Guide* for more details on using the -d startup option.

Problems with initialization of the TRMD include the following:

- Starting TRMD from the console

TRMD may fail with an ABEND=S000 U4093 REASON=00000090 because an OMVS segment was not defined for the TRMD ID.

Check the job output.

```
IEF403I TRMD - STARTED - TIME=12.48.55
ICH408I JOB(TRMD ) STEP(TRMD ) CL(PROCESS )
      OMVS SEGMENT NOT DEFINED
IEA995I SYMPTOM DUMP OUTPUT
      USER COMPLETION CODE=4093 REASON CODE=00000090
      TIME=12.48.58 SEQ=00065 CPU=0000 ASID=002B
      PSW AT TIME OF ERROR 078D1000 8000AA7A ILC 2 INTC 0D
      ACTIVE LOAD MODULE      ADDRESS=00007E70 OFFSET=0000
      NAME=CEEBINIT
      DATA AT PSW 0000AA74 - 00181610 0A0D47F0 B10A1811
      GR 0: 84000000 1: 84000FFD
      2: 00000090 3: 00000001
      4: 0001C2A0 5: 0001C144
      6: 00016560 7: 000169D0
      8: 00000016 9: 098E374E
      A: 00000004 B: 8000A9A8
      C: 00017AC0 D: 0001C018
      E: 00000000 F: 00000090
      END OF SYMPTOM DUMP
IEF450I TRMD TRMD - ABEND=S000 U4093 REASON=00000090
      TIME=12.48.58
IEF404I TRMD - ENDED - TIME=12.48.58
$HASP395 TRMD ENDED
```

CEE5101C During initialization, the OpenEdition callable service BPX1MSS failed. The system return code was 0000000156 , the reason code was 0B0C00F9 . The application will be terminated.

Verify that an OMVS segment exists for TRMD by issuing the `lu TSO` command from a user ID that has authority to issue the LU command: `LU trmd OMVS`. If an OMVS segment does not exist, use the ALU command to update the user's OMVS data. For example, `ALTUSER trmd OMVS(UID(0000) HOME('/') PROGRAM('/bin/sh'))`.

- The TCP/IP stack is not up. Message EZZ8498I is received.
Verify that the TCP/IP stack is up.

Documentation for the IBM Software Support Center

When contacting the IBM Software Support Center for problem resolution, some or all of the following information may be required:

- Gather TRMD debugging data by starting TRMD with command `trmd -d 3`. See “Diagnosing TRMD problems” on page 578.
- Start CTRACE in the stack to gather related information. See “Component trace” on page 53.
- The output from the `pasearch -i` command. Refer to *z/OS Communications Server: IP System Administrator's Commands*.
- The output from the `NETSTAT IDS/onetstat -k` command. Refer to *z/OS Communications Server: IP System Administrator's Commands*.

Chapter 24. Diagnosing OROUTED problems

This chapter provides information and guidance to diagnose OROUTED problems.

Overview

The route daemon is a server that implements the routing information protocol (RIP) described in RFC 1058 (RIP Version 1) and in RFC 1723 (RIP Version 2). It provides an alternative to the static TCP/IP gateway definitions. When configured properly, the MVS host running with OROUTED becomes an active RIP router in a TCP/IP network. The OROUTED server dynamically creates and maintains the network routing tables using RIP. RIP allows gateways and routers to periodically broadcast their routing tables to adjacent nodes. This enables the OROUTED server to update the host routing table. For example, the OROUTED server can determine if a new route has been created, if a route is temporarily unavailable, or if a more efficient route exists.

Before OROUTED was implemented for TCP/IP, static route tables were used for routing IP datagrams over connected networks. However, the static routes had a drawback in that they were not able to respond to changes in the network. By implementing the Routing Information Protocol (RIP) between a host and TCP/IP, the OROUTED server dynamically updates the internal routing tables whenever changes to the network occur.

The OROUTED server reacts to network topology changes on behalf of TCP/IP by maintaining the host routing tables, processing and generating RIP datagrams, and performing error recovery procedures.

Figure 65 on page 582 shows the OROUTED environment.

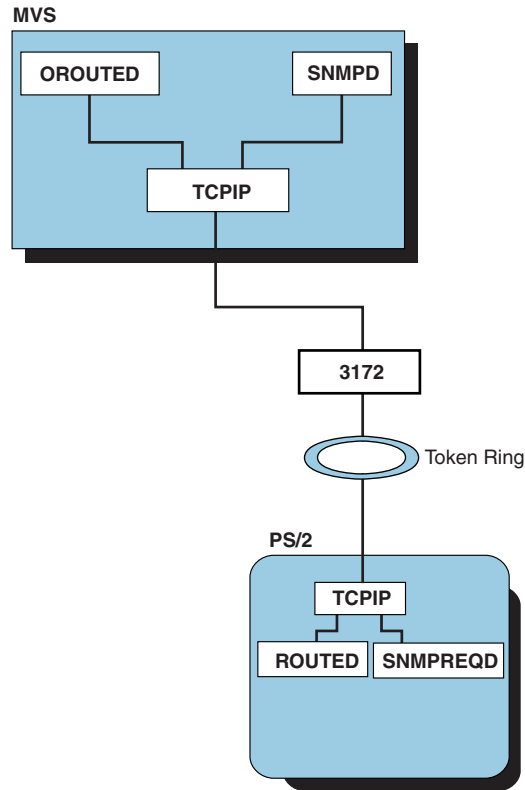


Figure 65. OROUTED environment

The OROUTED protocol is based on the exchange of RIP messages. There are two types of messages:

Request message

Sent from a client (another RIP router) as a request to transmit all or part of this host routing table.

Response message

Sent from OROUTED to a client (another RIP router) containing all or part of this host routing table.

Definitions

OROUTED must be defined correctly to TCP/IP, and it needs to be started by a RACF authorized user ID. Refer to the *z/OS Communications Server: IP Configuration Reference* for detailed information about configuring OROUTED and TCP/IP definitions.

Diagnosing OROUTED problems

Problems with OROUTED are generally reported under one of the following categories:

- “Abends” on page 583
- “OROUTED connection problems” on page 583
- “Analyzing routing failures” on page 584
- “Incorrect output” on page 586
- “Session outages” on page 586

Use the information provided in the following sections for problem determination and diagnosis of errors reported against OROUTED.

Abends

An abend during OROUTED processing should result in messages and error-related information being sent to the system console. A dump of the error is needed unless the symptoms match a known problem.

OROUTED connection problems

OROUTED connection problems are reported when OROUTED is unable to connect to TCP/IP. Generally, this type of problem is caused by an error in the configuration or definitions in TCP/IP.

In a CINET environment (multiple stacks), OROUTED attempts to connect to a stack whose name is determined by the TCPIPjobname keyword from the resolver configuration data set or file. If OROUTED cannot determine the TCPIPjobname, it will use a default TCPIPjobname of INET. If OROUTED cannot communicate with the stack pointed to by TCPIPjobname, it ends and issues an error message. A copy of OROUTED must be started for each stack requiring OROUTED services.

In configurations with multiple stacks, a copy of OROUTED must be started for each stack requiring OROUTED services. To associate OROUTED with a particular stack, use the environment variable RESOLVER_CONFIG to point to the data set or file that defines the unique TCPIPjobname. Refer to *z/OS Communications Server: IP Configuration Reference* for additional information about the search order for the TCPIP.DATA data set.

OROUTED and OMPROUTE cannot run on the same stack concurrently.

Documentation

The following documentation should be available for initial diagnosis of OROUTED connection problems:

- PROFILE.TCPIP information
- TCPIP.DATA information
- OROUTED cataloged procedure, if used
- MVS system log
- OROUTED standard output and standard error
- The file (or data set) pointed to by the GATEWAYS_FILE environmental variable
- The file (or data set) pointed to by the RESOLVER_CONFIG environmental variable
- The file (or data set) pointed to by the ROUTED_PROFILE environmental variable

Analysis

Refer to the *z/OS Communications Server: IP Configuration Reference* for TCP/IP configuration-related problems.

Diagnostic steps for OROUTED connection problems:

1. Verify that the TCP/IP job name is correct in the resolver configuration data set or file. Remember, if you start OROUTED from a cataloged procedure, the file on the STDENV statement might define the RESOLVER_CONFIG environmental variable.
2. If starting from a cataloged procedure, make sure you are using the appropriate cataloged z/OS UNIX application for OROUTED. Verify the correctness of the data set references.

3. Make sure that OROUTED is configured correctly in the PROFILE.TCPIP information. UDP port 520 must be reserved for OROUTED.
4. Make sure that OROUTED is configured correctly in the services file or data set. Verify that the assigned port number and service name are correct.
5. For network connectivity problems, see Chapter 4, “Diagnosing network connectivity problems” on page 25.

Analyzing routing failures

Routing problems are usually the result of outages in a network and the lack of alternative routing paths available for recovery. They can also be the result of incorrect configurations in the channel-attached and network-attached routers, as well as incorrect ARP entries when applicable. PING and Traceroute commands to and from a z/OS or OS/390 host are useful diagnosis aids for problem determination. In this section, unless otherwise specified, the following command terms are used:

PING

Refers to z/OS UNIX oping, TSO PING, and the ping commands used on other platforms.

Traceroute

Refers to z/OS UNIX otracert/traceroute, TSO TRACERTE, and the traceroute commands used on other platforms

NETSTAT ROUTE

Refers to the z/OS UNIX onetstat -r, TSO NETSTAT ROUTE, and the netstat route commands used on other platforms.

NETSTAT GATE

Refers to the z/OS UNIX onetstat -g and TSO NETSTAT GATE commands. This command is available only on the z/OS platform.

NETSTAT ARP

Refers to the z/OS UNIX onetstat -R ALL, TSO NETSTAT ALL, and the netstat arp commands used on other platforms.

If a PING or Traceroute command fails on a system where OROUTED is being used, a client is unable to get a positive response to a PING or Traceroute command. Before doing anything else, do the following:

1. If using a name for the destination host, try using an IP address. If the IP address works, then the problem is related to DNS configuration. See Chapter 32, “Diagnosing resolver problems” on page 683 for problem determination.
2. Issue the NETSTAT ROUTE commands on the local and remote hosts to get the routing table information.
3. Issue the TRACERT command to the destination host. If the destination host is on a local multi-access network, issue the NETSTAT ARP command to determine if there is an ARP entry for the remote host. If there is a TRANSLATE statement defined in the TCPIP profile for this host, make sure that its MAC address is correct.

From the NETSTAT ROUTE outputs, determine which route is used to reach the destination and determine the route active state. In general, a routing table is searched in the following order, starting with most specific to the least specific:

1. Host Routes
2. Subnet Routes

3. Network Routes
4. Supernet Routes
5. Default Routes

If the local or remote is a z/OS host, the NETSTAT GATE command can be issued to display the MTUs and subnet masks for the routes in the routing table.

If there are no active routes available to reach the destination or if there are channel-attached or network-attached routers along the routing path that are not configured properly, the PING and TRACERT commands will fail. If the routes are shown to be active at the local and remote hosts, the problem is most likely to be caused by a router along the routing path. Use the output from the TRACERT command to locate the suspect router.

Documentation

The following documentation should be available for initial diagnosis of routing problems:

- MVS system log
- PROFILE.TCPIP information
- Output from NETSTAT ROUTE commands for both local and remote hosts.
- Output from NETSTAT GATE command if the local and remote is a z/OS host.
- Output from MODIFY OROUTED, TABLES command for OROUTED's internal routing table display.
- Outputs from PING and TRACERT commands.
- Outputs from NETSTAT ARP commands.

Analysis

Follow the steps below to diagnose routing problems:

1. Issue the TRACERT loopback command. This will determine whether or not the command is working.
2. Issue the TRACERT command to the remote host. Make sure that the command contains a valid destination IP address for the remote host.

If the destination IP address is a virtual IP address (VIPA), make sure that VIPA is defined correctly. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information on rules and recommendations when defining a virtual IP address.
3. Make sure that the router providing the RIP support involved in the PING or TRACERT transaction is active and is running with a correct level of OROUTED or some application that provides RIP support.

If a channel-attached or network-attached router is not running RIP for the local network and is used to reach a remote host, make sure that passive or static routes are defined to and from the router so that the remote host can be reachable.
4. If the PING or TRACERT command was issued from a remote host, issue the NETSTAT ROUTE, NETSTAT GATE (if host is z/OS), and NETSTAT ARP commands from there for its routing and ARP table information. Verify that the routes and networks are correct as defined in the *hlq.PROFILE.TCPIP* data set and the OROUTED gateways file.
5. From DOS, issue **IFCONFIG inet ip show** to display the TCP/IP configured routes.

For routers or hosts running on platforms other than z/OS, refer to their documentation for more information about how to correct routing problems. Also, refer to these documentations for NETSTAT commands to display the routing and ARP tables for problem determination.

6. If there are no problems with the routes and networks, check for broken or poorly-connected cables between the client and the remote host. This includes checking the intranet interfaces (such as token ring and Ethernet) on the OROUTED server. Also, if the local or remote host is attached to an OSA device, ensure that its configuration is correct. Refer to the OSA configuration documentation for more information.

Incorrect output

Problems with incorrect output are reported when the data sent to the client is in an un expected form (for example, incorrect TCP/IP output, invalid RIP commands, incorrect RIP broadcasting information, incorrect routing table updates, or truncated packets).

Documentation

The following documentation should be available for initial diagnosis of incorrect output:

- OROUTED cataloged procedure, if used
- MVS system log
- OROUTED standard output and standard error
- Output from MODIFY OROUTED, TABLES command for OROUTED's internal routing table display
- PROFILE.TCPIP information
- Outputs from NETSTAT GATE and NETSTAT ROUTE commands.

Analysis

Table 33 shows types of incorrect output and describes the actions needed for initial diagnosis of the error.

Table 33. OROUTED incorrect output

Incorrect output	Action steps
TCP/IP incorrect output	<ol style="list-style-type: none">1. If the TCP/IP console shows a TCP/IP error message, refer to <i>z/OS Communications Server: IP Messages Volume 1 (EZA)</i> and follow the directions for system programmer response for the message.2. In the event of TCP/IP loops or hangs, see Chapter 3, "Diagnosing abends, loops, and hangs" on page 21.
OROUTED incorrect output	If an OROUTED error message is displayed, refer to <i>z/OS Communications Server: IP Messages Volume 1 (EZA)</i> and follow the directions for system programmer response for the message.

Session outages

Session outages are reported as an unexpected abend or termination of a TCP/IP connection.

Documentation

The following documentation should be available for initial diagnosis of session outages:

- OROUTED cataloged procedure, if used
- MVS system log
- OROUTED standard output and standard error
- Output from MODIFY OROUTED, TABLES command for OROUTED's internal routing table display

- TCPIP CTRACE (see Chapter 5, “TCP/IP services traces and IPCS support” on page 53)
- PROFILE.TCPIP information
- Outputs from NETSTAT GATE and NETSTAT ROUTE commands.

Analysis

Table 34 shows types of session outages and describes the steps needed for initial diagnosis of the error.

Table 34. OROUTED session outages

Session outage	Action steps
TCP/IP session outage	<ol style="list-style-type: none"> 1. If the TCP/IP console shows a TCP/IP error message, refer to <i>z/OS Communications Server: IP Messages Volume 1 (EZA)</i> and follow the directions for system programmer response for the message. 2. Examine the external CTRACE for information about the error. 3. In the event of an TCP/IP abend, see Chapter 3, “Diagnosing abends, loops, and hangs” on page 21.
OROUTED session outage	If an OROUTED error message is displayed, refer to <i>z/OS Communications Server: IP Messages Volume 1 (EZA)</i> and follow the directions for system programmer response for the message.

OROUTED traces and debug information

There are many TCP/IP traces that can be useful in identifying the cause of OROUTED problems. OROUTED traces and debug requests can be started from the z/OS UNIX shell, or they can be started from an MVS cataloged procedure. This section discusses both methods.

Note: OROUTED trace output is sent to syslogd unless you specify otherwise. See “Where to send OROUTED trace output” on page 590.

Figure 66 on page 588 shows a sample OROUTED environment.

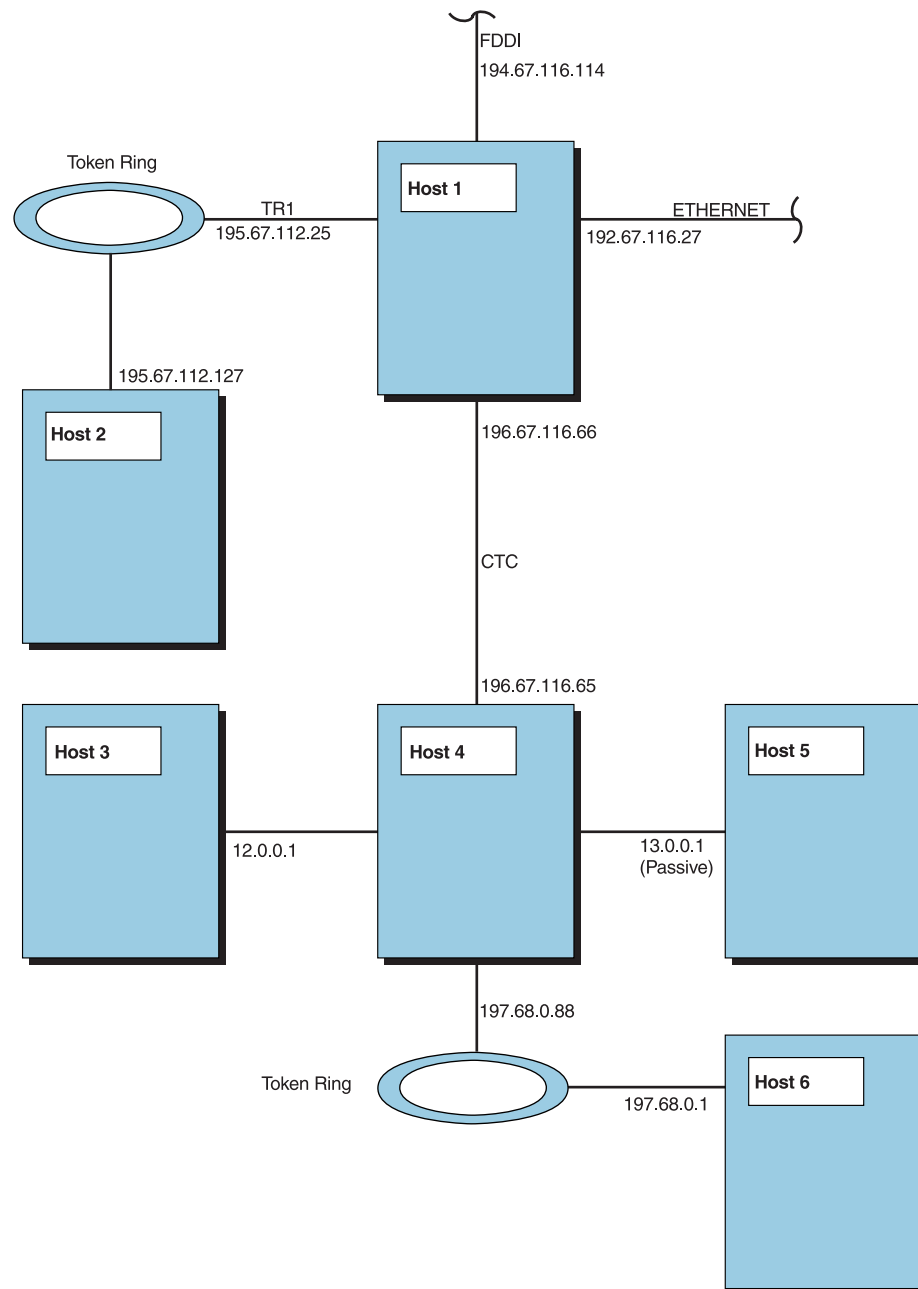


Figure 66. Sample OROUTED environment

Starting OROUTED traces from the z/OS UNIX shell

From TSO, issue the OMVS command, which puts you at a UNIX-like prompt. Then type the orouted command followed by one or more of the following parameters:

Note: OROUTED traces can be dynamically started and stopped using the MODIFY command. For more information, refer to the *z/OS Communications Server: IP Configuration Reference*.

-t Activates tracing of actions by the OROUTED server.

- t -t** Activates tracing of actions and packets sent or received.
- t -t -t** Activates tracing of actions, packets sent or received, and packet history. Circular trace buffers are used for each interface to record the history of all packets traced. This history is included in the trace output whenever an interface becomes inactive.
- t -t -t -t** Activates tracing of actions, packets sent or received, packet history, and packet contents. The RIP network routing information is included in the trace output.
- dp** Activates tracing of packets to and from adjacent routers and received and broadcasted RIP network routing tables. Packets are shown in data format in the trace output. The information is written to standard output.
- ep** Sends print statement to standard output and standard error.
- d** Enables internal debug information, which consists of internal code points. The information is written to standard output. Use this parameter only if IBM service requests the information.

Notes:

1. The parameters described here include only those that activate tracing. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about all of the OROUTED parameters.
2. To run orouted in the background, add an ampersand (&) to the command, as in the command `orouted &`
3. You can enter more than one parameter. Use a space after each parameter (for example, `orouted -t -t -t -dp.`)

Starting OROUTED traces from an MVS catalogued procedure

The OROUTED traces are controlled by parameters on PARM= in the PROC statement of the OROUTED cataloged procedure.

For example:

```
//OROUTED EXEC PGM=OROUTED,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON)',
// 'ENVAR("_CEE_ENVFILE=DD:STDENV")/-ep -t -t')
```

The OROUTED parameters that control tracing are:

Note: OROUTED traces can be dynamically started and stopped using the MODIFY command. For more information, refer to the *z/OS Communications Server: IP Configuration Reference*.

- t** Activates tracing of actions by the OROUTED server.
- t -t** Activates tracing of actions and packets sent or received.
- t -t -t** Activates tracing of actions, packets sent or received, and packet history. Circular trace buffers are used for each interface to record the history of all packets traced. This history is included in the trace output whenever an interface becomes inactive.
- t -t -t -t** Activates tracing of actions, packets sent or received, packet history, and packet contents. The RIP network routing information is included in the trace output.

- dp** Activates tracing of packets to and from adjacent routers and received and broadcasted RIP network routing tables. Packets are shown in data format in the trace output. The information is written to standard output.
- ep** Sends print statement to standard output and standard error.
- d** Enables internal debug information, which consists of internal code points. The information is written to standard output. Use this parameter only if IBM service requests the information.

Notes:

1. Each parameter is separated by a blank.
2. Parameters can be specified in mixed case.
3. The parameters described here include only those that activate tracing. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about all of the OROUTED parameters.

Where to send OROUTED trace output

Normally, trace output is sent to syslogd, which includes a great deal of information about many TCP/IP components, not just OROUTED. If you want to see only OROUTED output, use the **-ep** parameter when you start a trace. This sends output to syslogd and to standard output.

If you are running in the z/OS UNIX shell, using the **-ep** parameter sends the output to the shell session screen. You can save this output by redirecting it into a file with the greater-than (>) character (for example, `orouted -ep -t -t > orouted.stdout`).

Note: The **-ep** parameter cannot be altered using the MODIFY command.

Stopping OROUTED

You can stop OROUTED in several ways:

- From a z/OS UNIX shell superuser ID, issue the `kill` command to the process ID (PID) associated with OROUTED.

To find the PID, use one of the following methods:

- Use `D OMVS,U=USERID`. (This is the USERID that started OROUTED from the shell.)
- Use the `ps -ef` command from the shell.
- Write down the PID when you start OROUTED.
- Use a shell command pipeline such as
`kill $(ps | awk '/orouted/' {print $1})`

(In this case, you do not need to know the PID.)

- From MVS, issue the MODIFY command, specifying the parameter **-k** or **-kdr**. The **-k** parameter stops OROUTED, while the **-kdr** parameter stops OROUTED and deletes all dynamic routes. For example, the following command stops an OROUTED server started with a procedure named OROUTED.

```
MODIFY OROUTED,PARMS=-k
```

For more information on this command, refer to the *z/OS Communications Server: IP Configuration Reference*.

- From MVS, issue `P procname` where *procname* is the procedure name used to start OROUTED. If OROUTED was started from the z/OS UNIX shell, the

procname is *useridX*, where *X* is the sequence number set by the system. To determine the sequence number, issue `/d a,l` from any MVS console, to see the programs running.

Note: To see the z/OS UNIX application name in the output, use `D OMVS,U=USERID`, as described previously.

Changing trace and debug levels with MODIFY

Whether you start OROUTED from OMVS or MVS, you can use the MVS MODIFY to change command trace levels. The MODIFY syntax is

```
MODIFY procname,parms=parm
```

A modify format for MVS might look like this:

```
MODIFY OROUTED,PARMS=-t -t -t
```

If you had started from the z/OS UNIX shell, you would use something like this:

```
MODIFY useridX,PARMS=-t -t -t
```

where *X* is the sequence number for the OROUTED job. To determine the sequence number, see “Stopping OROUTED” on page 590.

The `-tq` parameter disables all traces.

You can also turn on debug information with the parameters `-d`, `-dp`, or both. To turn off debug information, use the `-dq` parameter.

For more information on using the MODIFY command, refer to the *z/OS Communications Server: IP Configuration Reference*.

OROUTED trace example and explanation

Figure 67 on page 592 shows an example of an OROUTED trace that was generated using `-ep -t -t -t` parameters. Short descriptions of numbered items in the trace follow the figure.

Note: Unless specifically shown otherwise, all examples in this chapter are valid for V1R2.

```

EZZ4990I OE Routed server initializing. Level 10.00
EZZ4980I Using catalog '/usr/lib/nls/msg/C/routed.cat' for OE Routed messages.
(1) EZZ4828I Input parameter(s): -ep -t -t -t
EZZ4985I Setting High Level Qualifier (HLQ) to 'CS390'
EZZ4988I OE Routed established affinity with 'TCPCS'
EZZ4929I Port 520 assigned to route
EZZ4932I *****
EZZ5001I Opening OE Routed profile /u/user156/r.p
EZZ4932I *****
EZZ5002I RIP_SUPPLY_CONTROL: RIP2M
EZZ4932I *****
EZZ4850I * Processing interface TR1
EZZ4932I *****
EZZ4948I This interface is not point-to-point
EZZ4943I Adding network route for interface
EZZ4882I Tue Jan 18 14:25:32 2000:
EZZ4883I ADD destination 9.0.0.0, router 9.67.116.94, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL timer 0
EZZ4943I Adding subnetwork route for interface
EZZ4883I ADD destination 9.67.116.0, router 9.67.116.94, metric 1
flags UP state INTERFACE|CHANGED|SUBNET timer 0
EZZ4932I *****
EZZ4850I * Processing interface VLINK1
EZZ4932I *****
EZZ4965I Virtual interface
EZZ4948I This interface is not point-to-point
EZZ4943I Adding network route for interface
EZZ4883I ADD destination 94.0.0.0, router 94.94.94.94, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL|VIRTUAL timer 0
EZZ4943I Adding subnetwork route for interface
EZZ4883I ADD destination 94.94.94.92, router 94.94.94.94, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL|SUBNET|VIRTUAL timer 0
EZZ4943I Adding host route for interface
EZZ4883I ADD destination 94.94.94.94, router 94.94.94.94, metric 1
flags UP|HOST state INTERFACE|CHANGED|INTERNAL|VIRTUAL timer 0
EZZ4932I *****
EZZ4850I * Processing interface CTC00
EZZ4932I *****
EZZ4940I Point-to-point interface, using broadaddr
EZZ4949I Interface CTC00 not up
EZZ4932I *****
EZZ4850I * Processing interface CTC02
EZZ4932I *****
EZZ4940I Point-to-point interface, using dstaddr
EZZ4943I Adding network route for interface
EZZ4883I ADD destination 215.215.215.0, router 215.215.215.2, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL timer 0
EZZ4943I Adding host route for interface
EZZ4883I ADD destination 215.215.215.4, router 215.215.215.2, metric 1
flags UP|HOST state INTERFACE|CHANGED timer 0

```

Figure 67. Example of an OROUTED trace (Part 1 of 7)

```

EZZ4932I *****
EZZ4850I * Processing interface VIPL0B0B0B0B
EZZ4932I *****
EZZ4965I Virtual interface
EZZ4948I This interface is not point-to-point
EZZ4943I Adding network route for interface
EZZ4883I ADD destination 11.0.0.0, router 11.11.11.11, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL|VIRTUAL timer 0
EZZ4943I Adding subnetwork route for interface
EZZ4883I ADD destination 11.11.11.0, router 11.11.11.11, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL|SUBNET|VIRTUAL timer 0
EZZ4943I Adding host route for interface
EZZ4883I ADD destination 11.11.11.11, router 11.11.11.11, metric 1
flags UP|HOST state INTERFACE|CHANGED|INTERNAL|VIRTUAL timer 0
EZZ4932I *****
EZZ4850I * Processing interface EZASAMEMVS
EZZ4932I *****
EZZ4940I Point-to-point interface, using broadaddr
EZZ4943I Adding network route for interface
EZZ4883I ADD destination 150.150.0.0, router 150.150.150.1, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL timer 0
EZZ4943I Adding subnetwork route for interface
EZZ4883I ADD destination 150.150.150.0, router 150.150.150.1, metric 1
flags UP state INTERFACE|CHANGED|SUBNET timer 0
EZZ4932I *****
EZZ4934I * Opening GATEWAYS file (/u/user156/g.f)
EZZ4932I *****
EZZ4925I Start of GATEWAYS processing:
EZZ4945I ifwithnet: compare with TR1
EZZ4947I netmatch 9.67.116.88 and 9.67.116.94
EZZ4936I Adding passive net route 0.0.0.0 via gateway 9.67.116.88, metric 1
EZZ4883I ADD destination 0.0.0.0, router 9.67.116.88, metric 1
flags UP|GATEWAY state PASSIVE|CHANGED|DEFAULT timer 0
EZZ5013I RIP2 authentication disabled at interface level (TR1)
EZZ5013I RIP2 authentication disabled at interface level (VLINK1)
EZZ5013I RIP2 authentication disabled at interface level (CTC00)
EZZ5019I Joining multicast group 224.0.0.9 on interface CTC00
EZZ5013I RIP2 authentication disabled at interface level (CTC02)
EZZ5019I Joining multicast group 224.0.0.9 on interface CTC02
EZZ5013I RIP2 authentication disabled at interface level (VIPL0B0B0B0B)
EZZ5013I RIP2 authentication disabled at interface level (EZASAMEMVS)
EZZ5019I Joining multicast group 224.0.0.9 on interface EZASAMEMVS
EZZ4926I End of GATEWAYS processing
EZZ4849I OE Routed Server started
(2) EZZ4899I REQUEST to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:25:32 2000
EZZ4949I Interface CTC00 not up
EZZ4899I REQUEST to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4899I REQUEST to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:25:32 2000
(3) EZZ4829I Waiting for incoming packets
EZZ4899I REQUEST from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4899I REQUEST from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4899I RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4899I REQUEST from 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:25:32 2000

```

Figure 67. Example of an OROUTED trace (Part 2 of 7)

```

EZZ4958I supply 9.67.116.94 -> 520 via TR1
EZZ4899I     RESPONSE to 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:25:32 2000
EZZ4899I     RESPONSE from 9.67.116.88 -> 520: ver 1 Tue Jan 18 14:25:32 2000
EZZ4882I Tue Jan 18 14:25:32 2000:
EZZ4883I ADD destination 9.67.113.0, router 9.67.116.88, metric 2
flags UP|GATEWAY state CHANGED|SUBNET timer 0
EZZ4883I ADD destination 9.67.114.0, router 9.67.116.88, metric 2
flags UP|GATEWAY state CHANGED|SUBNET timer 0
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 9.67.116.88 -> 520: ver 1 Tue Jan 18 14:25:50 2000
EZZ4829I Waiting for incoming packets
(4) EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4957I 30 second timer expired (broadcast)
EZZ4958I supply 9.67.116.255 -> 0 via TR1
EZZ4899I     RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:26:02 2000
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:26:02 2000
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:26:02 2000
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:26:02 2000
EZZ4899I     RESPONSE from 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:26:02 2000
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:26:02 2000
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 9.67.116.88 -> 520: ver 1 Tue Jan 18 14:26:19 2000
EZZ4829I Waiting for incoming packets
EZZ4957I 60 second timer expired (rescan kernel for interfaces)
EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4949I Interface CTC00 not up
EZZ4957I 30 second timer expired (broadcast)
EZZ4958I supply 9.67.116.255 -> 0 via TR1
EZZ4899I     RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:26:33 2000
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:26:33 2000
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:26:33 2000
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:26:33 2000
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:26:33 2000
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:26:33 2000
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
(5) EZZ4828I Input parameter(s): -T -T -T -T
EZZ4871I Tracing packet contents enabled Tue Jan 18 14:26:46 2000
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 9.67.116.88 -> 520: ver 1 Tue Jan 18 14:26:49 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 1
EZZ4902I     destination 9.67.114.0 metric 1

```

Figure 67. Example of an OROUTED trace (Part 3 of 7)

```

EZZ4829I Waiting for incoming packets
EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4957I 30 second timer expired (broadcast)
EZZ4958I supply 9.67.116.255 -> 0 via TR1
EZZ4899I     RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 2
EZZ4902I     destination 9.67.114.0 metric 2
EZZ4902I     destination 9.67.116.0 metric 1
EZZ4902I     destination 150.150.150.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 2
EZZ4902I     destination 9.67.114.0 metric 2
EZZ4902I     destination 9.67.116.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 2
EZZ4902I     destination 9.67.114.0 metric 2
EZZ4902I     destination 9.67.116.0 metric 1
EZZ4902I     destination 150.150.150.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 2
EZZ4902I     destination 9.67.114.0 metric 2
EZZ4902I     destination 9.67.116.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4899I     RESPONSE from 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.0.0.0 metric 1

```

Figure 67. Example of an OROUTED trace (Part 4 of 7)

```

EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
EZZ4957I 60 second timer expired (rescan kernel for interfaces)
EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4949I Interface TR1 not up
(6) EZZ4891I *** Packet history for interface TR1 ***
EZZ4898I Output trace:
EZZ4899I REQUEST to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:25:32 2000
EZZ4903I request for full tables
EZZ4899I RESPONSE to 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:25:32 2000
(7) EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 150.150.0.0 metric 1
EZZ4902I destination 215.215.215.0 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4899I RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:26:02 2000
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 150.150.0.0 metric 1
EZZ4902I destination 215.215.215.0 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4899I RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:26:33 2000
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 150.150.0.0 metric 1
EZZ4902I destination 215.215.215.0 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4899I RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:27:03 2000
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 150.150.0.0 metric 1
EZZ4902I destination 215.215.215.0 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4892I *** End packet history ***
EZZ4882I Tue Jan 18 14:27:33 2000:
EZZ4889I CHANGE metric destination 9.0.0.0, router 9.67.116.94, from 1 to 16
EZZ4889I CHANGE metric destination 9.67.113.0, router 9.67.116.88, from 2 to 16
EZZ4889I CHANGE metric destination 9.67.114.0, router 9.67.116.88, from 2 to 16
EZZ4889I CHANGE metric destination 9.67.116.0, router 9.67.116.94, from 1 to 16
EZZ4916I deleting route to interface TR1 (timed out)
EZZ4949I Interface CTC00 not up
EZZ4957I 30 second timer expired (broadcast)
EZZ4949I Interface TR1 not up
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:27:33 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 1
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 150.150.150.0 metric 1
EZZ4902I destination 150.150.0.0 metric 1
EZZ4902I destination 94.94.94.92 metric 1
EZZ4902I destination 94.0.0.0 metric 1

```

Figure 67. Example of an OROUTED trace (Part 5 of 7)

```

EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:27:33 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 1
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 215.215.215.0 metric 1
EZZ4902I destination 94.94.94.92 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4829I Waiting for incoming packets
EZZ4899I RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:27:33 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 1
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 150.150.150.0 metric 1
EZZ4902I destination 150.150.0.0 metric 1
EZZ4902I destination 94.94.94.92 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4899I RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:27:33 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 1
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 215.215.215.0 metric 1
EZZ4902I destination 94.94.94.92 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4957I 30 second timer expired (broadcast)
EZZ4949I Interface TR1 not up
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:28:03 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 1
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 150.150.150.0 metric 1
EZZ4902I destination 150.150.0.0 metric 1
EZZ4902I destination 94.94.94.92 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:28:03 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 1
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 215.215.215.0 metric 1
EZZ4902I destination 94.94.94.92 metric 1
EZZ4902I destination 94.0.0.0 metric 1

```

Figure 67. Example of an OROUTED trace (Part 6 of 7)


```

EZZ4829I Waiting for incoming packets
EZZ4899I RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:28:03 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 1
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 150.150.150.0 metric 1
EZZ4902I destination 150.150.0.0 metric 1
EZZ4902I destination 94.94.94.92 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4899I RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:28:03 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 1
EZZ4902I destination 11.0.0.0 metric 1
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 215.215.215.0 metric 1
EZZ4902I destination 94.94.94.92 metric 1
EZZ4902I destination 94.0.0.0 metric 1
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
(5) EZZ4828I Input parameter(s): -KDR
EZZ4949I Interface TR1 not up
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:28:08 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 16
EZZ4902I destination 11.0.0.0 metric 16
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 150.150.150.0 metric 16
EZZ4902I destination 150.150.0.0 metric 16
EZZ4902I destination 94.94.94.92 metric 16
EZZ4902I destination 94.0.0.0 metric 16
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:28:08 2000
EZZ4902I destination 9.0.0.0 metric 16
EZZ4902I destination 11.11.11.0 metric 16
EZZ4902I destination 11.0.0.0 metric 16
EZZ4902I destination 9.67.113.0 metric 16
EZZ4902I destination 9.67.114.0 metric 16
EZZ4902I destination 9.67.116.0 metric 16
EZZ4902I destination 215.215.215.0 metric 16
EZZ4902I destination 94.94.94.92 metric 16
EZZ4902I destination 94.0.0.0 metric 16

```

Figure 67. Example of an OROUTED trace (Part 7 of 7)

Following are short descriptions of the numbered items in the trace:

- (1) EZZ4828I shows the parameters used for this trace.
- (2) EZZ4899I describes the type of packet being sent or received.
- (3) EZZ4829I means the system is waiting for incoming packets.
- (4) EZZ4957I shows timer information.
- (5) At this point, trace parameters were changed using a MODIFY command.
- (6) EZZ4891I shows packet history.

(7) EZZ4902I provides packet detail for each route contained in an RIP packet.

Documentation for the IBM Software Support Center

When contacting the IBM Software Support Center for problem resolution, some or all of the following information may be required:

- PROFILE.TCPIP information
- Output from `onetstat -g` and `onetstat -r`
- Network diagram or layout
- OROUTED profile (the file pointed to by the `ROUTED_PROFILE` environmental variable)
- OROUTED gateways file (the file pointed to by the `GATEWAYS_FILE` environmental variable)
- Standard output and standard error of OROUTED using the `-ep` parameter with full tracing on (for example, running `orouted -ep -t -t -t -t`)

Chapter 25. Diagnosing OMPROUTE problems

This chapter provides information and guidance to diagnose OMPROUTE problems.

Overview

OMPROUTE implements the Open Shortest Path First (OSPF) protocol described in RFC 1583 (OSPF Version 2) as well as the Routing Information Protocols (RIP) described in RFC 1058 (RIP Version 1) and in RFC 1723 (RIP Version 2).

OMPROUTE provides an alternative to the static TCP/IP gateway definitions. When configured properly, the MVS host running with OMPROUTE becomes an active OSPF and/or RIP router in a TCP/IP network. Either (or both) of these two routing protocols can be used to dynamically maintain the host routing table. For example, OMPROUTE can determine that a new route has been created, that a route is temporarily unavailable, or that a more efficient route exists.

OMPROUTE has the following characteristics:

- It is a z/OS UNIX application. It requires the Hierarchical File System (HFS) to operate.
- OMPROUTE can be started from an MVS procedure, from the OS/390 shell, or from AUTOLOG. Refer to the *z/OS Communications Server: IP Configuration Guide* for information about OMPROUTE.
- The OMPROUTE subagent provides an alternative to DISPLAY commands for displaying Open Shortest Path First (OSPF) protocol configuration and state information. The subagent implements the Management Information Base (MIB) variables defined in Request for Comment (RFC) 1850. The OMPROUTE subagent is controlled by statements in the OMPROUTE configuration file. For details, refer to the *z/OS Communications Server: IP Configuration Reference*.
- OMPROUTE needs to be started by a RACF authorized user ID.
- OMPROUTE needs to be in an APF authorized library.
- A one-to-one relationship exists between an instance of OMPROUTE and a TCP/IP stack.
- OSPF/RIP support on multiple TCP/IP stacks using the split-stack function requires multiple instances of OMPROUTE.
- OMPROUTE and OROUTED cannot run on the same TCP/IP stack concurrently.
- All dynamic routes are deleted from the routing table upon initialization of OMPROUTE.
- Internet Control Message Protocol (ICMP) redirects are ignored when OMPROUTE is active.
- Unlike OROUTED, OMPROUTE does not make use of the BSD Routing Parameters. Instead, the Maximum Transmission Unit (MTU), subnet mask, and destination address parameters are configured using the OSPF_Interface, RIP_Interface, and Interface statements in the OMPROUTE configuration file.
- OMPROUTE uses the MVS operator console, SYSLOGD, STDOUT, and CTRACE for its logging and tracing.
 - The MVS operator console and SYSLOGD are used for major events such as initialization, termination, and error conditions.
 - STDOUT is used for detailed tracing and debugging.
 - CTRACE is used for the following purposes:
 - Tracing the receipt and transmission of OSPF/RIP packets

- Tracing subagent/SNMP agent packets
- Communication between OMPROUTE and the TCP/IP stack

For details on using TCP/IP Services Component trace support with OMPROUTE, see “TCP/IP services component trace for OMPROUTE” on page 615 and Chapter 5, “TCP/IP services traces and IPCS support” on page 53.

- If you want to communicate a routing protocol over an interface, configure the interface to OMPROUTE using the OSPF_INTERFACE or RIP_INTERFACE configuration statement.
- Interfaces that are not involved in the communication of the RIP or OSPF protocol (except VIPA interfaces) must be configured to OMPROUTE using the INTERFACE configuration statement, unless it is a non-point-to-point interface and all default values are acceptable as specified on the INTERFACE statement. For details, refer to the *z/OS Communications Server: IP Configuration Guide*.
- OMPROUTE uses a standard message catalog. The message catalog must be in the HFS. The directory location for the message catalog path is set by the environment variables NLSPATH and LANG.
- OMPROUTE is enhanced with Virtual IP Addressing (VIPA) to handle network interface failures by switching to alternate paths. The virtual routes are included in the OSPF and RIP advertisements to adjacent routers. Adjacent routers learn about virtual routes from the advertisements and can use them to reach the destinations at the MVS host.
- OMPROUTE allows for the generation of multiple, equal-cost routes to a destination, thus providing load-balancing support.

OMPROUTE works best without non-replaceable static routes, and the use of non-replaceable static routes (defined using the GATEWAY TCP/IP configuration statement) is not recommended. Non-replaceable static routes may interfere with the discovery of a better route to the destination as well as inhibit the ability to switch to another route if the destination should become unreachable by way of the static route. For example, if you define a host route through one interface and that interface becomes unreachable, OMPROUTE does not acknowledge your static route and does not define a host route through an alternate interface.

If you must define replaceable static routes, all replaceable static routes will be considered to be of equal cost and replaceable static routes will not be replaced by OSPF or RIP routes. Use extreme care when working with static routes and OMPROUTE. Set IMPORT_STATIC_ROUTES = YES on the AS_Boundary Routing configuration statement or set SEND_STATIC_ROUTES = YES on the RIP_Interface configuration statement if you want the static routes to be advertised to other routers.

You can define static routes as replaceable. Unlike non-replaceable static routes, replaceable static routes will always be replaced by dynamic routes learned by OMPROUTE. In other words, a replaceable static route will be used only if no dynamic route is known to the destination. Replaceable static routes can be thought of as last resort routes to reach a destination when no dynamic route is known.

Definitions

OMPROUTE must be defined correctly to TCP/IP. For detailed information about TCP/IP definitions, refer to the chapter on configuring OMPROUTE in the *z/OS Communications Server: IP Configuration Reference*.

Diagnosing OMPROUTE problems

Problems with OMPROUTE are generally reported under one of the following categories:

- Abends
- OMPROUTE connection problems
- Routing failures

These categories are described in the following sections.

Abends

An abend during OMPROUTE processing should result in messages and error-related information being sent to the system console. A dump of the error is needed unless the symptoms match a known problem.

OMPROUTE connection problems

OMPROUTE connection problems are reported when OMPROUTE is unable to connect to TCP/IP or to one of the ports required for OSPF or RIP communication. Generally, an inability to connect to TCP/IP is caused by an error in the configuration or definitions in TCP/IP. An inability to connect to one of the required ports is generally caused by an error in the configuration or definitions in TCP/IP or by attempting to start OMPROUTE when either OMPROUTE or OROUTED is already connected to the specified stack.

In a Common INET environment (multiple stacks), OMPROUTE attempts to connect to a stack whose name is determined by the TCPIPjobname keyword in the resolver configuration data set or file. If OMPROUTE cannot determine the TCPIPjobname, it uses a default of INET.

If OMPROUTE cannot communicate with the stack pointed to by TCPIPjobname or is unable to initialize its required ports, it issues an error message describing the problem and then terminates.

For details on diagnosing problems while attempting to connect to the SNMP agent, see “SNMP connection problems” on page 492.

Routing failures

Routing problems are usually the result of outages in a network and a lack of alternative routing paths available for recovery. They can also be the result of incorrect configurations in the channel-attached and network-attached routers as well as incorrect ARP entries when applicable. PING and Traceroute commands to and from a z/OS host are useful diagnosis aids for problem determination. In this section, unless otherwise specified, the following command terms are used:

PING

Refers to z/OS UNIX oping, TSO PING, and the ping commands used on other platforms.

Traceroute

Refers to z/OS UNIX otracertltraceroute, TSO TRACERTE, and the traceroute commands used on other platforms

NETSTAT ROUTE

Refers to the z/OS UNIX onetstat -r, TSO NETSTAT ROUTE, and the netstat route commands used on other platforms.

NETSTAT GATE

Refers to the z/OS UNIX onetstat -g and TSO NETSTAT GATE commands. This command is only available on the z/OS platform.

NETSTAT ARP

Refers to the z/OS UNIX onetstat -R ALL, TSO NETSTAT ALL, and the netstat arp commands used on other platforms.

If a PING or Traceroute command fails on a system where OMPROUTE is being used, a client is unable to get a positive response to a PING or TRACERT command. Before doing anything else, do the following:

1. If using a name for the destination host, try using a IP address. If the IP address works, then the problem is related to DNS configuration. See Chapter 32, “Diagnosing resolver problems” on page 683 for problem determination.
2. Issue the NETSTAT ROUTE commands on the local and remote hosts to get the routing table information.
3. Issue the TRACERT command to the destination host. If the destination host is located on a local multi-access network, issue the NETSTAT ARP command to determine if there is an ARP entry for the remote host. If there is TRANSLATE statement defined in the TCPIP profile for this host, make sure that its MAC address is correct.

From the NETSTAT ROUTE outputs, determine which route is used to reach the destination and determine the route active state. In general, a routing table is searched in the following order, starting with most specific to the least specific:

1. Host Routes
2. Subnet Routes
3. Network Routes
4. Supernet Routes
5. Default Routes

If the local or remote is a z/OS host, the NETSTAT GATE command can be issued to display the MTUs and subnet masks for the routes in the routing table.

If there are no active routes available to reach the destination or if there are improperly configured channel-attached or network-attached routers along the routing path, the PING and Traceroute commands will fail. To function correctly, PING requires active routes in both directions between the Ping origin and the Ping destination. If the routes are shown to be active at the local and remote hosts, the problem is most likely to be caused by a router along the routing path. Use the output from the Traceroute command to locate the suspect router.

Documenting routing failures

The following documentation should be available for initial diagnosis of routing failures:

- MVS system log.
- Output from onetstat -r or TSO NETSTAT ROUTE command.
- SYSLOGD.
- The data set containing OMPROUTE trace and debug information. For details, see “OMPROUTE traces and debug information” on page 605.
- TCP/IP and OMPROUTE CTRACE. For information about generating an OMPROUTE Component Trace, see “TCP/IP services component trace for OMPROUTE” on page 615.

- Output from appropriate OMPROUTE DISPLAY commands as described in the *z/OS Communications Server: IP Configuration Reference*.
- Output from NETSTAT ROUTE commands for both local and remote hosts.
- Output from NETSTAT GATE command if the local and remote is a z/OS host.
- Outputs from PING and Traceroute commands.
- Outputs from NETSTAT ARP commands.

Analyzing routing failures

When analyzing routing failures, follow these guidelines:

- Make sure that the address used in attempting to contact the remote host is a valid IP address.
- If the output from the `onetstat -r` or TSO NETSTAT ROUTE command does not show the expected results relative to the desired destination, do one or more of the following:
 - Make sure routing is possible in both directions. For most TCP/IP communication, two-way routing is required. The origin must have routes to reach the destination, and the destination must have routes to reach the origin. So if the netstat done at the origin shows correct routing, you must also do netstat route displays at the destination to verify that it can send replies back to the origin.

Also, this is affected by SOURCEVIPA. If SOURCEVIPA is enabled at the origin of the communication, then the destination must be able to route back to the VIPA address that the origin will use as its source address. If there are intermediate hops between the source and destination, all routing tables must have routing information. For example, if the origin node routing table indicates that the first hop to reach the destination is router A, then the router A routing table must also have a valid, active route to the destination, and so on. This also applies to the return route.
 - Make sure that the router or routers involved in providing information relative to this destination are operational and participating in the correct routing protocol.
 - Make sure that the physical connections involved in reaching the destination are active.
 - Use the OMPROUTE DISPLAY commands described in the *z/OS Communications Server: IP Configuration Reference* to determine if anything in the configuration or current state of OMPROUTE has resulted in the absence of a route to the destination.

OMPROUTE traces and debug information

There are many TCP/IP traces that can be useful in identifying the cause of OMPROUTE problems. OMPROUTE's use of the MVS Component Trace support is also useful (see "TCP/IP services component trace for OMPROUTE" on page 615). This section describes the OMPROUTE internal traces. OMPROUTE internal tracing and debugging can be started when OMPROUTE is started. Also, the MODIFY command can be used to start, stop, or alter OMPROUTE tracing and debugging after OMPROUTE has been started.

This section describes each of these methods.

Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell

If OMPROUTE is started from the z/OS UNIX System Services shell command line (using the `omproute` command), parameters can be specified to indicate the level of tracing or debugging desired.

- **-tn (where n is a supported trace level)**

This option specifies the external tracing level. It is intended for customers, testers, service, or developers, and provides information on the operation of the routing application. This option can be used for many purposes, such as debugging a configuration, education on the operation of the routing application, verification of testcases, and so on. The following trace levels are supported:

- 1 = Informational messages
- 2 = Formatted packet trace

- **-sn (where n is a supported debug level)**

This option specifies the internal debugging level for the OMPROUTE subagent. It is intended for service or developers only and provides internal debugging information needed for debugging problems. The following level is supported:

- 1 = Internal debugging messages. Turns on `DPIdebug(2)`.

- **-dn (where n is a supported debug level)**

This option specifies the internal debugging level. It is intended for service or developers only and provides internal debugging information needed for debugging problems. The following levels are supported:

- 1 = Internal debugging messages
- 2 = Unformatted hex packet trace
- 3 = Function entry or exit trace
- 4 = Task add or run

Notes:

1. The `-tn`, `-sn`, and `-dn` options affect OMPROUTE performance. As a result, you may have to increase the Dead Router Interval on OSPF interfaces to prevent neighbor adjacencies from collapsing.
2. The trace and debug levels are cumulative; each level includes all lower levels. For example, `-t2` provides formatted packet trace and informational messages. You can enter more than one parameter by inserting a space after each parameter; for example, `omproute -t1 -d2`.
3. Parameters can be specified in mixed case.

Starting OMPROUTE tracing and debugging from an MVS cataloged procedure or AUTOLOG

The OMPROUTE tracing and debugging are controlled by parameters on `PARM=` when OMPROUTE is started from an MVS cataloged procedure or AUTOLOG. For example:

```
//OMPRROUTE EXEC PGM=BPXBATCH,REGION=4096K,TIME=NOLIMIT,  
//          PARM='PGM /usr/sbin/omproute -t1 -d2'
```

For a description of the parameters that can be specified, see “Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell”.

Starting OMPROUTE tracing and debugging using the MODIFY command

Whether you start OMPROUTE from the z/OS UNIX System Services shell or from a MVS cataloged procedure, you can use the MODIFY command to start logging or tracing, to stop logging or tracing, and to change the level of logging or tracing. The syntax for these MODIFY commands follows:

- **MODIFY *procname*,TRACE=*trace-level***
Use the TRACE command to change the trace level that may have been set as a start option. This command is intended for customers, testers, service, or developers.
 - TRACE=0 turns off OMPROUTE tracing.
 - TRACE=1 gives all the informational messages.
 - TRACE=2 gives the informational messages plus formatted packet tracing.
- **MODIFY *procname*,DEBUG=*debug-level***
Use the DEBUG command to change the debug level that may have been set as a start option. This command is intended for service or developers only.
 - DEBUG=0 turns off OMPROUTE debugging.
 - DEBUG=1 gives internal debug messages.
 - DEBUG=2 gives the same as DEBUG=1 plus hexadecimal packet tracing.
 - DEBUG=3 gives the same as DEBUG=2 plus module entry and exit.
 - DEBUG=4 gives the same as DEBUG=3 plus task add and run.
- **MODIFY *procname*,SADEBUG=*trace-level***
Use the SADEBUG command to start and stop message logging for the OMPROUTE subagent and to stop DPI tracing:
 - SADEBUG=0 stops message logging for the OMPROUTE subagent and issues DPidebug(0) to stop DPI tracing.
 - SADEBUG=1 generates all messages by the OMPROUTE subagent and DPidebug(2).

Destination of OMPROUTE trace and debug output

Output from OMPROUTE tracing and debugging is written to the debug output destination. When using the MODIFY command with debug or trace turned on, the output destination is based on the OMPROUTE_DEBUG_FILE environment variable. If OMPROUTE was started with tracing enabled and OMPROUTE_DEBUG_FILE is not defined, the trace output destination is stdout. If OMPROUTE was started without tracing enabled and OMPROUTE_DEBUG_FILE is not defined, the trace output destination is *\$TMP/omproute_debug*, where *\$TMP* is the value of the TMP environment variable.

When OMPROUTE_DEBUG_FILE is defined, the first trace file created will be the value coded on OMPROUTE_DEBUG_FILE. When that file is full, the extension will be changed to 00N, where N is in the range of 1 to the number of files specified in the OMPROUTE_DEBUG_CONTROL environment variable (default 4). The current file is always the value defined as OMPROUTE_DEBUG_FILE and the oldest file is the highest N value. This eliminates the danger of OMPROUTE filling the HFS when tracing is active for a long time.

The size and number of debug files created can be controlled by the OMPROUTE_DEBUG_FILE_CONTROL environment variable. This allows you to adjust how much data OMPROUTE traces. You tailor this parameter to your network complexity or available HFS storage capacity, or network complexity and

available HFS storage capacity. Refer to the *z/OS Communications Server: IP Configuration Guide* for details on this environment variable.

Sample OMPROUTE trace output

Figure 68 on page 609 is a sample OMPROUTE trace with descriptions for some of the trace entries:

```

1  EZZ7800I OMROUTE starting
   EZZ7889I 00 ACTIVE COMP=SYSTCPRT SUB=USER1464
   EZZ7845I Established affinity with TCPCS8
   EZZ7817I Using defined OSPF protocol 89
   EZZ7838I Using configuration file: /u/user146/omproute/omproute.conf
2  EZZ7883I Processing interface from stack, address 9.169.100.18,
   name CTC2, index 2, flags 451
   EZZ7883I Processing interface from stack, address 9.67.100.8,
   name CTC1, index 1, flags 451
   EZZ8023I The RIP routing protocol is Enabled
   EZZ7937I The OSPF routing protocol is Enabled
3  EZZ8057I Added network 9.67.100.0 to interface 9.67.100.8
   on net 0 interface CTC1
   EZZ7827I Adding stack route to 9.67.100.0, mask 255.255.255.0 via
   0.0.0.0, link CTC1, metric 1, type 1
   EZZ8057I Added network 9.67.100.7 to interface 9.67.100.8 on net 0
   interface CTC1
   EZZ7827I Adding stack route to 9.67.100.7, mask 255.255.255.255 via
   0.0.0.0, link CTC1, metric 1, type 129
4  EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
   interface CTC1
   EZZ7879I Joining multicast group 224.0.0.5 on interface 9.67.100.8
5  EZZ7913I State change, interface 9.67.100.8, new state 16,
   event 1

:
EZZ7875I No default route defined
EZZ8100I OMROUTE subagent Starting
EZZ7898I OMROUTE Initialization Complete
EZZ8101I OMROUTE subagent Initialization Completed
EZZ7908I Received packet type 1 from 9.167.100.13
6  EZZ8011I send request to address 9.67.100.7
   EZZ8015I sending packet to 9.67.100.7
   EZZ8011I send request to address 9.169.100.14
   EZZ8015I sending packet to 9.169.100.14
   EZZ8015I sending packet to 9.67.100.7
   EZZ8012I sending broadcast response to address 9.67.100.255 in 1
   packets with 1 routes
   EZZ8015I sending packet to 9.169.100.14
   EZZ8012I sending broadcast response to address 9.169.100.255 in 1
   packets with 1 routes
7  EZZ7908I Received packet type 1 from 9.67.100.7
   EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
   interface CTC1
8  EZZ7919I State change, neighbor 9.67.100.7, new state 4, event 1
9  EZZ7919I State change, neighbor 9.67.100.7, new state 8, event 3
   EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8
   org 9.67.100.8
10 EZZ7919I State change, neighbor 9.67.100.7, new state 16,
   event 14
11 EZZ7910I Sending multicast, type 2, destination 224.0.0.5 net
   0 interface CTC1
12 EZZ7908I Received packet type 2 from 9.67.100.7
13 EZZ7919I State change, neighbor 9.67.100.7, new state 32, event 5
14 EZZ7910I Sending multicast, type 3, destination 224.0.0.5 net 0
   interface CTC1
   EZZ7908I Received packet type 2 from 9.67.100.7
15 EZZ7908I Received packet type 4 from 9.67.100.7

```

Figure 68. Sample OMROUTE Trace Output (Part 1 of 6)

```

16 EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id
    9.67.100.7 org 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id 9.67.100.8
    org 9.67.100.8
EZZ7927I from 9.67.100.7, self update: typ 1 id 9.67.100.8 org
    9.67.100.8
EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id
    9.167.100.13 org 9.100.13
EZZ7928I from 9.67.100.7, new LS advertisement: typ 5 id 9.67.100.0
    org 9.67.100.8
EZZ7927I from 9.67.100.7, self update: typ 5 id 9.67.100.0 org
    9.67.100.8
EZZ7928I from 9.67.100.7, new LS advertisement: typ 5 id 9.169.100.0
    org 9.67.100.8
EZZ7927I from 9.67.100.7, self update: typ 5 id 9.169.100.0 org
    9.67.100.8
EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org
    9.67.100.8
17 EZZ7910I Sending multicast, type 4, destination 224.0.0.5 net
    0 interface CTC1
EZZ7910I Sending multicast, type 3, destination 224.0.0.5 net 0
    interface CTC1
EZZ7908I Received packet type 4 from 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 5 id
    9.169.100.14 org 9.67.100.8
EZZ7927I from 9.67.100.7, self update: typ 5 id 9.169.100.14 org
    9.67.100.8
EZZ7910I Sending multicast, type 2, destination 224.0.0.5 net 0
    interface CTC1
EZZ7908I Received packet type 2 from 9.67.100.7
18 EZZ7919I State change, neighbor 9.67.100.7, new state 128,
    event 6
19 EZZ7908I Received packet type 5 from 9.67.100.7
20 EZZ7910I Sending multicast, type 5, destination 224.0.0.5
    net 0 interface CTC1
EZZ8015I sending packet to 9.169.100.14
EZZ8012I sending broadcast response to address 9.169.100.255 in
    1 packets with 1 routes
EZZ8015I sending packet to 9.67.100.7
EZZ8012I sending broadcast response to address 9.67.100.255 in
    1 packets with 1 routes
EZZ8015I sending packet to 9.169.100.14
EZZ8012I sending broadcast response to address 9.169.100.255 in
    1 packets with 1 routes
EZZ7908I Received packet type 4 from 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id
    9.67.100.7 org 9.67.100.7
EZZ7910I Sending multicast, type 5, destination 224.0.0.5 net 0
    interface CTC1
EZZ7934I Originating LS advertisement: typ 5 id 9.169.100.14 org
    9.67.100.8
EZZ7934I Originating LS advertisement: typ 5 id 9.169.100.0 org
    9.67.100.8
EZZ7934I Originating LS advertisement: typ 5 id 9.67.100.0 org
    9.67.100.8
EZZ7910I Sending multicast, type 4, destination 224.0.0.5 net 0
    interface CTC1

```

Figure 68. Sample OMPROUTE Trace Output (Part 2 of 6)

```

21 EZZ7949I Dijkstra calculation performed, on 2 area(s)
EZZ7935I New OMPROUTE route to destination Net 9.67.100.7,
    type SPF cost 1
EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.7 org
    9.67.100.8
EZZ7908I Received packet type 5 from 9.67.100.7
EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org
    9.67.100.8
EZZ7910I Sending multicast, type 4, destination 224.0.0.5 net 0
    interface CTC1
EZZ7908I Received packet type 4 from 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id
    9.167.100.13 org 9.167.100.13
EZZ7928I from 9.67.100.7, new LS advertisement: typ 4 id
    9.67.100.8 org 9.167.100.13
EZZ7928I from 9.67.100.7, new LS advertisement: typ 3 id
    9.67.100.7 org 9.167.100.13
EZZ7908I Received packet type 5 from 9.67.100.7
EZZ7910I Sending multicast, type 5, destination 224.0.0.5 net 0
    interface CTC1
EZZ7949I Dijkstra calculation performed, on 2 area(s)
22 EZZ7827I Adding stack route to 9.167.100.13, mask 255.255.
    255.255 via 9.67.100.7, link CTC1, metric 2, type 129
EZZ7935I New OMPROUTE route to destination Net 9.167.100.13,
    type SPF cost 2
EZZ7935I New OMPROUTE route to destination Net 9.67.100.8,
    type SPF cost 2
EZZ7913I State change, interface 9.67.100.8, new state 16, event 1
EZZ7935I New OMPROUTE route to destination BR 9.167.100.13,
    type SPF cost 2
EZZ7827I Adding stack route to 9.167.100.17, mask 255.255.255.255
    via 9.67.100.7, link CTC1, metric 3, type 129
EZZ7935I New OMPROUTE route to destination Net 9.167.100.17,
    type SPF cost 3
EZZ7934I Originating LS advertisement: typ 3 id 9.167.100.13 org
    9.67.100.8
EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.8 org
    9.67.100.8
EZZ7934I Originating LS advertisement: typ 3 id 9.167.100.17 org
    9.67.100.8
23 EZZ7909I Sending unicast type 1 dst 9.167.100.13
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
    interface CTC1
EZZ7908I Received packet type 1 from 9.167.100.13
EZZ7919I State change, neighbor 9.167.100.13, new state 4, event 1
EZZ7919I State change, neighbor 9.167.100.13, new state 8, event 3
EZZ7919I State change, neighbor 9.167.100.13, new state 16, event 14
EZZ7909I Sending unicast type 2 dst 9.167.100.13
EZZ7908I Received packet type 4 from 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 4 id
    9.67.100.8 org 9.167.100.13
EZZ7928I from 9.67.100.7, new LS advertisement: typ 3 id
    9.67.100.7 org 9.167.100.13
EZZ7908I Received packet type 2 from 9.167.100.13
EZZ7919I State change, neighbor 9.167.100.13, new state 32, event 5
EZZ7909I Sending unicast type 2 dst 9.167.100.13
EZZ7910I Sending multicast, type 5, destination 224.0.0.5 net 0
    interface CTC1
EZZ7908I Received packet type 2 from 9.167.100.13
EZZ7909I Sending unicast type 3 dst 9.167.100.13
EZZ7908I Received packet type 4 from 9.167.100.13

```

Figure 68. Sample OMPROUTE Trace Output (Part 3 of 6)

```

EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 1
        interface CTC2
EZZ7928I from 9.167.100.13, new LS advertisement: typ 1 id
        9.67.100.8 org 9.67.100.8
EZZ7927I from 9.167.100.13, self update: typ 1 id 9.67.100.8 org
        9.67.100.8
:
EZZ7909I Sending unicast type 4 dst 9.167.100.13
EZZ7919I State change, neighbor 9.167.100.13, new state 128, event 6
EZZ7909I Sending unicast type 2 dst 9.167.100.13
EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org
        9.67.100.8
EZZ7910I Sending multicast, type 4, destination 224.0.0.5 net 0
        interface CTC1
EZZ7933I Flushing advertisement: typ 3 id 9.67.100.7 org 9.167.100.13
EZZ7933I Flushing advertisement: typ 4 id 9.67.100.8 org 9.167.100.13
EZZ7909I Sending unicast type 5 dst 9.167.100.13
EZZ8015I sending packet to 9.67.100.7
EZZ8012I sending broadcast response to address 9.67.100.255 in
        1 packets with 1 routes
EZZ7908I Received packet type 5 from 9.67.100.7
EZZ7908I Received packet type 1 from 9.67.100.7
EZZ8004I response received from host 9.67.100.7
EZZ7908I Received packet type 5 from 9.167.100.13
EZZ7949I Dijkstra calculation performed, on 2 area(s)
EZZ8015I sending packet to 9.169.100.14
EZZ8012I sending broadcast response to address 9.169.100.255 in
        1 packets with 1 routes
EZZ7908I Received packet type 4 from 9.167.100.13
EZZ7928I from 9.167.100.13, new LS advertisement: typ 1 id
        9.167.100.13 org 9.167.100.13
EZZ7908I Received packet type 4 from 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id
        9.167.100.13 org 9.167.100.13
EZZ7910I Sending multicast, type 5, destination 224.0.0.5 net 0
        interface CTC1
EZZ7909I Sending unicast type 5 dst 9.167.100.13
EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org
        9.67.100.8
EZZ7909I Sending unicast type 4 dst 9.167.100.13
EZZ7908I Received packet type 5 from 9.167.100.13
EZZ8062I Subnet 9.0.0.0 defined
EZZ7949I Dijkstra calculation performed, on 2 area(s)
EZZ7935I New OMPROUTE route to destination BR 9.167.100.13,
        type SPF cost 2
24 EZZ7895I Processing DISPLAY command - OSPF,LIST,INTERFACES
EZZ7809I EZZ7833I INTERFACE CONFIGURATION
EZZ7809I IP ADDRESS      AREA      COST  RTRNS  TRNSDLY PRI  HELLO  DEAD
EZZ7809I 9.169.100.18      0.0.0.0    1    10      1    1    20    80
EZZ7809I 9.67.100.8        2.2.2.2    1    10      1    1    20    80
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
        interface CTC1
EZZ7908I Received packet type 1 from 9.167.100.13
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 1
        interface CTC2
EZZ7909I Sending unicast type 1 dst 9.167.100.13
EZZ7908I Received packet type 1 from 9.67.100.7
EZZ8015I sending packet to 9.67.100.7

```

Figure 68. Sample OMPROUTE Trace Output (Part 4 of 6)

```

EZZ8012I sending broadcast response to address 9.67.100.255 in
1 packets with 1 routes
EZZ8004I response received from host 9.67.100.7
EZZ8015I sending packet to 9.169.100.14
EZZ8012I sending broadcast response to address 9.169.100.255 in
1 packets with 1 routes
25 EZZ7895I Processing MODIFY command - TRACE=2
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
interface CTC1
26 EZZ7876I -- OSPF Packet Sent ----- Type: Hello
EZZ7878I OSPF Version: 2 Packet Length: 48
EZZ7878I Router ID: 9.67.100.8 Area: 2.2.2.2
EZZ7878I Checksum: 1dcf Authentication Type: 0
EZZ7878I Hello_Interval: 20 Network mask: 255.255.255.0
EZZ7878I Options: E
EZZ7878I Router_Priority: 1 Dead_Router_Interval: 80
EZZ7878I Backup_DR: 0.0.0.0 Designated_Router: 0.0.0.0
EZZ7878I Neighbor: 9.67.100.7
EZZ7877I -- OSPF Packet Received -- Type: Hello
EZZ7878I OSPF Version: 2 Packet Length: 48
EZZ7878I Router ID: 9.67.100.7 Area: 2.2.2.2
EZZ7878I Checksum: 1dcf Authentication Type: 0
EZZ7878I Hello_Interval: 20 Network mask: 255.255.255.0
EZZ7878I Options: E
EZZ7878I Router_Priority: 1 Dead_Router_Interval: 80
EZZ7878I Backup_DR: 0.0.0.0 Designated_Router: 0.0.0.0
EZZ7878I Neighbor: 9.67.100.8
EZZ7908I Received packet type 1 from 9.67.100.7
27 -- RIP Packet Received -- Type: Response (V1)
Destination_Addr: 9.169.100.0 metric: 2
EZZ8004I response received from host 9.67.100.7
-- RIP Packet Sent ----- Type: Response (V1)
Destination_Addr: 9.169.100.0 metric: 1
EZZ8015I sending packet to 9.67.100.7
EZZ8012I sending broadcast response to address 9.67.100.255 in
1 packets with 1 routes
28 EZZ7895I Processing MODIFY command - TRACE=1
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 1
interface CTC2
EZZ7909I Sending unicast type 1 dst 9.167.100.13
EZZ7908I Received packet type 1 from 9.67.100.7
EZZ8004I response received from host 9.67.100.7
EZZ8015I sending packet to 9.67.100.7
EZZ8004I response received from host 9.67.100.7
EZZ8015I sending packet to 9.67.100.7
EZZ8012I sending broadcast response to address 9.67.100.255 in 1
packets with 1 routes
EZZ8015I sending packet to 9.169.100.14
EZZ8012I sending broadcast response to address 9.169.100.255 in 1
packets with 1 routes
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
interface CTC1
:
EZZ7909I Sending unicast type 1 dst 9.167.100.13
EZZ7908I Received packet type 1 from 9.67.100.7
29 EZZ7862I Received update interface CTC1

```

Figure 68. Sample OMPROUTE Trace Output (Part 5 of 6)

```

30 EZZ8061I Deleted net 9.67.100.0 route via 9.67.100.8 net 0
    interface CTC1
EZZ7864I Deleting all stack routes to 9.67.100.0, mask 255.255.255.0
31 EZZ7919I State change, neighbor 9.67.100.7, new state 1, event 11
EZZ7879I Leaving multicast group 224.0.0.5 on interface 9.67.100.8
32 EZZ7913I State change, interface 9.67.100.8, new state 1, event 7
EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org
    9.67.100.8
EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org
    9.67.100.8
EZZ7909I Sending unicast type 4 dst 9.167.100.13
EZZ8015I sending packet to 9.169.100.14
EZZ8012I sending broadcast response to address 9.169.100.255 in
    1 packets with 1 routes
EZZ7934I Originating LS advertisement: typ 5 id 9.67.100.0 org
    9.67.100.8
EZZ7933I Flushing advertisement: typ 5 id 9.67.100.0 org 9.67.100.8
EZZ7949I Dijkstra calculation performed, on 1 area(s)
EZZ7801I Deleting stack route to 9.67.100.7, mask 255.255.255.255
    via 0.0.0.0, link CTC1, metric 1, type 129
EZZ7935I New OMPROUTE route to destination Net 9.67.100.7,
    type SPIA cost 5
EZZ7943I Destination Net 9.167.100.13 now unreachable
EZZ7864I Deleting all stack routes to 9.167.100.13, mask
    255.255.255.255
EZZ7935I New OMPROUTE route to destination Net 9.67.100.8,
    type SPIA cost 4
EZZ7919I State change, neighbor 9.167.100.13, new state 1, event 11
EZZ7913I State change, interface 9.67.100.8, new state 1, event 7
EZZ7943I Destination BR 9.167.100.13 now unreachable
EZZ7943I Destination Net 9.167.100.17 now unreachable
EZZ7864I Deleting all stack routes to 9.167.100.17, mask
    255.255.255.255
EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.7
    org 9.67.100.8
EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.8 org
    9.67.100.8
:
EZZ7933I Flushing advertisement: typ 3 id 9.167.100.17 org 9.67.100.8
EZZ7933I Flushing advertisement: typ 3 id 9.67.100.8 org 9.67.100.8
EZZ7933I Flushing advertisement: typ 3 id 9.167.100.13 org 9.67.100.8
EZZ7933I Flushing advertisement: typ 3 id 9.67.100.7 org 9.67.100.8
EZZ8015I sending packet to 9.169.100.14
EZZ8012I sending broadcast response to address 9.169.100.255 in 1
    packets with 1 routes
EZZ7949I Dijkstra calculation performed, on 1 area(s)
EZZ7943I Destination Net 9.67.100.7 now unreachable
EZZ7943I Destination Net 9.67.100.8 now unreachable
EZZ7943I Destination BR 9.167.100.13 now unreachable
EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.7 org
    9.67.100.8
EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.8 org
    9.67.100.8
EZZ7933I Flushing advertisement: typ 3 id 9.67.100.8 org 9.67.100.8
EZZ7933I Flushing advertisement: typ 3 id 9.67.100.7 org 9.67.100.8
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 1
    interface CTC2
EZZ7804I OMPROUTE exiting

```

Figure 68. Sample OMPROUTE Trace Output (Part 6 of 6)

Following are brief explanations of numbered items in the trace:

- 1** OMPROUTE initializing (trace level 1 was specified at startup).

- 2** OMROUTE learns of TCP/IP stack interfaces.
- 3** Direct routes are added for each TCP/IP stack interface.
- 4** OSPF Hello packet sent out OSPF interface.
- 5** OSPF Interface transitions to state "point-to-point."
- 6** RIP Requests & Responses begin being sent out RIP interface.
- 7** OSPF Hello packet received from OSPF neighbor.
- 8** OSPF neighbor transitions to state "Init."
- 9** OSPF neighbor transitions to state "2-Way."
- 10** OSPF neighbor transitions to state "ExStart."
- 11** OSPF Database Description packet sent out OSPF interface.
- 12** OSPF Database Description received from OSPF neighbor.
- 13** OSPF neighbor transitions to state "Exchange."
- 14** OSPF Link State Request packet sent out OSPF interface.
- 15** OSPF Link State Update packet received from OSPF neighbor.
- 16** Link State Advertisements from received Update packet are processed.
- 17** OSPF Link State Update packet sent out OSPF interface.
- 18** OSPF neighbor transitions to state "Full."
- 19** OSPF Link State Acknowledgment packet received from OSPF neighbor.
- 20** OSPF Link State Acknowledgment packet sent out OSPF interface.
- 21** OSPF Dijkstra calculation is performed.
- 22** Learned route is added to TCP/IP stack route table.
- 23** Adjacency establishment begins with router at other end of OSPF Virtual Link.
- 24** Request received to display OSPF Interface configuration information.
- 25** Request received to change tracing level to 2 (adds formatted packets).
- 26** Formatted OSPF packet.
- 27** Formatted RIP packet.
- 28** Request received to change tracing level back to 1.
- 29** OMROUTE learns of stopped TCP/IP interface.
- 30** Routes over stopped interface are deleted.
- 31** Neighbor over stopped interface transitions to state "Down."
- 32** Stopped interface transitions to state "Down."

TCP/IP services component trace for OMROUTE

z/OS CS provides Component Trace support for the OMROUTE application. This section describes how to specify OMROUTE trace and formatting options. For short descriptions of other tracing procedures, such as displaying trace status, see Chapter 5, "TCP/IP services traces and IPCS support" on page 53. For detailed descriptions, refer to the following books:

- *z/OS MVS Diagnosis: Tools and Service Aids* for information about Component Trace procedures
- *z/OS MVS Initialization and Tuning Reference* for information about the SYS1.PARMLIB member
- *z/OS MVS System Commands* for information about trace commands
- *z/OS MVS Programming: Authorized Assembler Services Guide* for information about procedures and return codes for CTRACE macros

Specifying trace options

You can specify Component Trace options at TCP/IP initialization or after TCP/IP has initialized.

Specifying options at initialization

A default minimum Component Trace is always started during OMPROUTE initialization. To customize the parameters used to initialize the trace, update the SYS1.PARMLIB member CTIORA00 (see Figure 69 on page 617). For a description of trace options, see Table 35 on page 618.

Note: Besides specifying the trace options, you can also change the OMPROUTE trace buffer size. The buffer size can be changed only at OMPROUTE initialization.

If the CTIORA00 member is not found when starting OMPROUTE, the following message is issued:

```
IEE5381 CTIORA00 MEMBER NOT FOUND in SYS1.PARMLIB
```

When this occurs, the OMPROUTE component trace is started with a buffer size of 1MB and the MINIMUM tracing option.

```

/*****
/*
/* TCP/IP for MVS
/* SMP/E Distribution Name: CTIORA00
/*
/* PART Name: CTIORA00
/*
/*
/* Copyright:
/* Licensed Materials - Property of IBM
/* 5694-A01
/* (C) Copyright IBM Corp. 1998,2001
/*
/*
/* Status: CSV1R2
/*
/*
/* DESCRIPTION = This parmlib member causes component trace for
/* the TCP/IP OMPROUTE application to be initialized
/* with a trace buffer size of 256K
/*
/* This parmlib members only lists those TRACEOPTS
/* value specific by OMPROUTE. For a complete list
/* of TRACEOPTS keywords and their values see
/* OS/390 MVS INITIALIZATION AND TUNING REFERENCE.
/*
/*
/*
/*
/*****
TRACEOPTS
/* ----- */
/* Optionally start external writer in this file (use both
/* WTRSTART and WTR with same wtr_procedure)
/* ----- */
/* WTRSTART(wtr_procedure)
/* ----- */
/* ON OR OFF: PICK 1
/* ----- */
/* ON
/* OFF
/* ----- */
/* BUFSIZE: A VALUE IN RANGE 128K TO 16M
/* CTRACE buffers reside in OMPROUTE Private storage
/* which is in the regions address space.
/* ----- */
/* BUFSIZE(256K)
/* WTR(wtr_procedure)
/* ----- */
/* OPTIONS: NAMES OF FUNCTIONS TO BE TRACED, OR "ALL"
/* ----- */
/* OPTIONS(
/* 'ALL '
/* 'MINIMUM '
/* 'ROUTE '
/* 'PACKET '
/* 'OPACKET '
/* 'RPACKET '
/* 'IPACKET '
/* 'SPACKET '
/* )

```

Figure 69. SYS1.PARMLIB member CTIORA00

Table 35 describes the available trace options.

Table 35. OMPROUTE trace options

Trace event	Description
ALL	Select all types of records. Be aware that this option slows performance.
MINIMUM	Select OMPROUTE's minimum level of tracing. Specifying MINIMUM is the same as specifying ROUTE.
ROUTE	Select information exchange and routing updates between the OMPROUTE application and the z/OS TCP/IP Services stack.
PACKET	Select all inbound and outbound packet flows. This is the same as specifying OPACKET, RPACKET, and IPACKET.
RPACKET	Select inbound and outbound packet flows for the RIP protocol.
OPACKET	Select outbound packets sent to z/OS TCP/IP with information regarding route or interface changes.
IPACKET	Select inbound packets sent from z/OS TCP/IP with information regarding route or interface changes.
SPACKET	Trace inbound and outbound packets sent between the SNMP agent and the OMPROUTE subagent.

Specifying options after initialization

After OMPROUTE initialization, you must use the TRACE CT command to change the component trace options. Each time a new Component Trace is initiated, all prior trace options are turned OFF and the new options are put into effect.

You can specify the trace options with or without the PARMLIB member. See Chapter 5, "TCP/IP services traces and IPCS support" on page 53.

Formatting OMPROUTE trace records

You can format component trace records using IPCS panels or a combination of the IPCS panels and the CTRACE command, either from a dump or from external-writer files. (For details, see Chapter 5, "TCP/IP services traces and IPCS support" on page 53.) Any combination of the following values can be entered as options to filter the CTRACE entries. The options must be entered using the format:

TYPE(option[,option]...)

- ROUTE
- OPACKET
- RPACKET
- IPACKET
- SPACKET

You cannot use the following as options when formatting OMPROUTE component traces:

- ALL
- MINIMUM
- PACKET

Chapter 26. Diagnosing NCPROUTE problems

The NCPROUTE protocol provides a standardized interface, through which a server program on one host (NCPROUTE) can manage the routing tables and respond to SNMP route table requests for another program (Network Control Program).

Figure 70 shows the NCPROUTE environment.

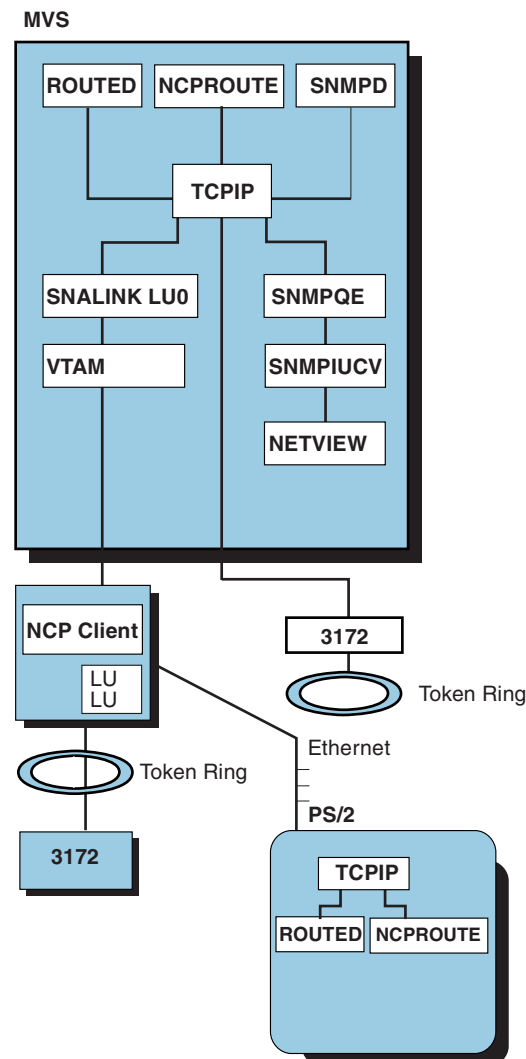


Figure 70. NCPROUTE environment

Prior to ACF/NCP V7R1, static route tables were used for routing IP datagrams over connected networks. However, the static routes had a drawback in that they were not able to respond to network topology changes. By implementing the RIP protocol between a host and NCP clients, the NCPROUTE server is able to provide dynamic IP routing for NCP clients. In effect, the NCP clients become active RIP routers in a TCP/IP network.

Multiple NCP units (374x family of communications controllers) can connect to the same NCPROUTE server on one host. This means that NCPROUTE can manage

multiple routing tables for each NCP client. SNALINK is used as the connection vehicle to establish LU0 sessions between NCPROUTE and NCP clients. Each NCP client can have one or more LU0 sessions with NCPROUTE, provided that one session is used as primary and others as secondary for backup.

The NCPROUTE server reacts to network topology changes on behalf of NCP clients by maintaining each NCP client routing table, processing and generating RIP and SNMP datagrams, and performing error recovery procedures.

The NCPROUTE protocol is based on the exchange of protocol data units (PDUs). There are eight types of PDUs:

- **Hello PDU:** Sent from an NCP client to initiate a session with NCPROUTE.
- **Acknowledge PDU:** Sent from NCPROUTE to acknowledge receipt of a Hello datagram. NCPROUTE is ready to manage the routing tables for an NCP client.
- **Status PDU:** Sent from an NCP client to inform NCPROUTE of a status change with an interface. Interfaces can become inactive or active.
- **Delete Route Request PDU:** Sent from NCPROUTE to request deletion of a route that is no longer known to the network from an NCP client routing table. This PDU can also be sent from an NCP client as a response informing NCPROUTE that the delete route request failed.
- **Add Route Request PDU:** Sent from NCPROUTE to request addition of a route that is discovered by NCPROUTE to an NCP client routing table. This PDU can also be sent from an NCP client as a response informing NCPROUTE that the add route request failed.
- **Change Route Request PDU:** Sent from NCPROUTE to request changing the value of a metric for a route currently active in an NCP client routing table.
- **Transport PDU:** Sent from an NCP client to request NCPROUTE to retransmit RIP broadcasts sent from other routers and to process Simple Network Management Protocol (SNMP) requests sent from SNMP clients in the network. This PDU can also be sent from NCPROUTE as a response to retransmit RIP broadcasts or as a response to an SNMP query request. The Transport PDU contains encapsulated RIP and SNMP commands for additional processing.
- **Inactive Interface List PDU:** Sent from an NCP client to inform NCPROUTE of currently inactive interfaces.

NCPROUTE uses the RIP commands for retransmitting of and responding to RIP broadcasts and trace requests. There are four types of RIP commands that can be encapsulated in a Transport PDU:

- **Request:** NCP received a request from a client (another RIP router) to retransmit RIP broadcasts.
- **Response:** Sent from NCP to its client (another RIP router) as a response to retransmit RIP broadcasts.
- **TraceOn:** NCP received a request from a client (another RIP router) to enable the actions trace provided by NCPROUTE.
- **TraceOff:** NCP received a request from a client (another RIP router) to disable tracing provided by NCPROUTE.

NCPROUTE communicates with the SNMP agent over the Distributed Program Interface (DPI) to process the SNMP commands. In this configuration, NCPROUTE becomes the SNMP subagent to provide values of registered MIB variables to the SNMP agent. There are four types of SNMP commands that can be encapsulated within a Transport PDU:

- **Get Request:** NCP received a request from a client to obtain one or more MIB variable values from an SNMP agent.
- **Get Next Request:** NCP received a request from a client to obtain the next variable value in the MIB tree from an SNMP agent.
- **Get Response:** Sent from NCP to its client as a response to an SNMP request.
- **Set Request:** NCP received a request from a client to set or change the value of one or more MIB variables in an SNMP agent. This command is not supported by NCPROUTE.

Refer to *z/OS Communications Server: IP User's Guide and Commands* for detailed information about the SNMP commands.

The following list describes the MIB variables registered for use by NCPROUTE:

- **ipRouteDest:** Destination IP address of this route
- **ipRouteMetric1:** Primary routing metric for this route
- **ipRouteMetric2:** Alternative routing metric for this route
- **ipRouteMetric3:** Another alternative routing metric for this route
- **ipRouteMetric4:** Another alternative routing metric for this route
- **ipRouteNextHop:** IP address of the next hop of this route
- **ipRouteType:** Type of route
- **ipRouteProto:** Routing mechanism by which this route was learned
- **ipRouteMask:** Mask value for this route

Refer to *z/OS Communications Server: IP User's Guide and Commands* for detailed information about the MIB variables.

Definitions

NCPROUTE must be defined correctly to both NCP and TCP/IP. NCPROUTE must be in the OBEY list of the *hlq.PROFILE.TCPIP* data set, and UDP port 580 must be reserved for NCPROUTE. Routes to the NCP clients must be defined on the GATEWAY or the BSDROUTINGPARMS statement for NCPROUTE connectivity.

Refer to the *z/OS Communications Server: IP Configuration Reference* for detailed information about TCP/IP and NCPROUTE server definitions.

Internet interfaces (token ring and Ethernet) and NCST logical units for communication with the TCP/IP host must be defined for each NCP client through NCP generation.

If you use SNMP to query routing information of NCP clients, the SNMP query engine and agent must be configured correctly. For NCPROUTE to communicate with the SNMP agent, the MVS host name or IP address and community name must be defined in the NCPROUTE profile, *hlq.SEZAINST(NCPRPROF)*. The SNMP agent community name must also be defined in the *hlq.PW.SRC* data set for proper verification.

Refer to the *z/OS Communications Server: IP Configuration Reference* for detailed information about SNMP definitions.

Diagnosing NCPROUTE problems

Problems with NCPROUTE are generally reported under one of the following categories:

- Abends
- Connection problems
- Analyzing routing failures
- Incorrect output
- Session outages

Use the information provided in the following sections for problem determination and diagnosis of errors reported against NCPROUTE.

Abends

An abend during NCPROUTE processing should result in messages and error related information being sent to the system console. A dump of the error is needed unless the symptoms match a known problem.

Documentation

Code a SYSUDUMP DD or SYSABEND DD statement in the cataloged procedure used to start NCPROUTE to ensure that a useful dump is obtained in event of an abend.

Analysis

Refer to *z/OS MVS Diagnosis: Procedures* or to Chapter 3, “Diagnosing abends, loops, and hangs” on page 21, in this manual for information about debugging dumps produced during NCPROUTE processing.

Connection problems

NCPROUTE connection problems are reported when NCPROUTE is unable to connect to TCP/IP, when NCP clients are unable to connect to the NCPROUTE server, when SNALINK LU0 is unable to connect between the NCPROUTE server and an NCP client, and when NCPROUTE is unable to connect to an SNMP agent. Generally, this type of problem is caused by an error in the configuration or definitions (either in VTAM, TCPIP, SNALINK, SNMP, NCP, or NCPROUTE).

Documentation

The following documentation should be available for initial diagnosis of NCPROUTE connection problems:

- Documentation for NCPROUTE connection failure
 - TCP/IP console log
 - *hlq*.PROFILE.TCPIP data set
 - TCPIP.DATA data set
 - NCPROUTE cataloged procedure
- Documentation for NCP client connection failure
 - NCPROUTE console log
 - NCPROUTE.PROFILE data set
 - NCP client network definitions data set (NCP generation)
- Documentation for SNALINK LU0 connection failure
 - SNALINK LU0 console log
 - VTAM APPL definitions for SNALINK LU0s
- Documentation for SNMP agent problems
 - SNMP console logs for SNMP agent and client
 - *hlq*.MIBDESC.DATA data set
 - *hlq*.PW.SRC data set
 - NetView log (if the SNMP client is on an MVS host)

More documentation that might be needed is discussed in the analysis section.

Analysis

Table 36 shows symptoms of connection problems and refers to the steps needed for initial diagnosis of the error.

Table 36. NCPROUTE connection problems

Connection Problem	Analysis Steps
NCP client connection failure	1, 2, 7, 8, 10, 14
NCPROUTE connection failure	1, 3, 5, 6, 7, 8, 10, 11, 14
SNALINK LU0 connection failure	1, 3, 7, 8, 10, 12
SNMP Agent connection failure	4, 9, 10, 13

Table 37 gives the diagnostic steps referred to in Table 36.

For TCP/IP configuration-related problems, refer to the *z/OS Communications Server: IP Configuration Reference* for more information.

Table 37. Diagnostic steps for NCPROUTE connection problems

Step	Action
1	For an NCP client, make sure that the internet interfaces (token ring and Ethernet) and NCST logical units for communication with the TCP/IP host are defined correctly in an NCP generation. Refer to the <i>ACF/NCP IP Router Planning and Installation Guide</i> for detailed information about NCP definitions.
1a	Make sure that the NCPROUTE UDP port (UDPPORT keyword), coded on the IPOWNER statement in an NCP generation, matches the value defined in the .ETC.SERVICES data set. If it is not coded, the value used will be the default UDP port 580.
1b	Verify that the assigned port numbers and service names for NCPROUTE and the router are correct. Also make sure that the router service port 520 is defined in the .ETC.SERVICES data set. The NCP clients will use this port as a destination port when broadcasting RIP packets to adjacent routers.
1c	Make sure that NCST logical units for the SNALINK LU0s are defined correctly. A partner LU name (INTERFACE keyword) for the SNALINK-NCST interface, coded on the LU statement in an NCST GROUP of an NCP generation, should match the LU name in a SNALINK LU0 DEVICE statement in the <i>hlq.PROFILE.TCPIP</i> data set.
1d	Make sure that the remote LU name (REMLU keyword) for the SNALINK-NCST interface, coded on the LU statement in an NCP generation, matches the VTAM application name in the VTAM APPL definitions for SNALINK LU0s. For more information about SNALINK configuration and VTAM APPL definitions, refer to the <i>z/OS Communications Server: IP Configuration Guide</i> .
1e	Make sure that the NCST partner LU name (INTERFACE keyword) for the SNALINK-NCST interface, coded on the IPOWNER and IPLOCAL statements in an NCP generation, matches the partner LU name in Step 1b.
1f	Make sure that the IP address for the TCP/IP host (HOSTADDR keyword), coded on the IPOWNER statement in an NCP generation, matches the IP address for the SNALINK LU0 device name coded on the HOME statement in the <i>hlq.PROFILE.TCPIP</i> data set.
1g	Make sure that the IP address for the SNALINK-NCST interface (LADDR keyword), coded on a IPLOCAL statement in an NCP generation, matches the IP address for the SNALINK LU0 link name coded on the GATEWAY statement in the <i>hlq.PROFILE.TCPIP</i> data set.

Table 37. Diagnostic steps for NCPROUTE connection problems (continued)

Step	Action
1h	Make sure that the destination IP address for the SNALINK-NCST interface (P2PDEST keyword), coded on a IPLOCAL statement in an NCP generation, matches the IP address on the IPOWNER statement in Step 1e.
1i	Make sure that IPLOCAL statements are defined for the directly-attached NCP internet interfaces (token ring and Ethernet) in an NCP generation. Verify the correctness of the IP addresses (LADDR keyword), metric values (METRIC keyword), protocol type (PROTOCOL keyword), and subnetwork masks (SNETMASK keyword).
2	Make sure that the appropriate NCP LOADLIB is used and that it contains correct network definitions. The NCP LOADLIB must be in the search list referred to by the //DD STEPLIB statement. Verify that a 374x communications controller to be in the session with NCPROUTE is loaded with the correct NCP load module.
3	Make sure that appropriate cataloged procedures for NCPROUTE (NCROUT) and SNALINK (SNALPROC) are used, and verify the correctness of the data set references.
3a	For the SNALINK cataloged procedure, make sure that the number of SNALINK sessions is large enough to allow multiple NCP sessions with NCPROUTE. This number is referred to by the MAXSESS keyword on the EXEC statement.
4	If using SNMP, make sure that the appropriate cataloged procedure for the SNMP agent (SNMPD) is used and verify the correctness of the data set references. Do likewise for a SNMP client (SNMPQE on MVS host).
5	Make sure that NCPROUTE is configured correctly in the <i>hlq</i> .PROFILE.TCPIP data set. The cataloged procedure name (NCROUT) is referred to on the OBEY, AUTOLOG (optional), and PORT statements. UDP port 580 must be reserved for NCPROUTE.
6	Make sure that NCPROUTE is configured correctly in the ETC.SERVICES data set. See also Step 1a.
7	Make sure that SNALINK LU0 is configured correctly in the <i>hlq</i> .PROFILE.TCPIP data set. The SNALINK device name, LU name, and VTAM application address space name are referred to on the DEVICE statement. The SNALINK link name is referred to on the LINK, HOME, and GATEWAY statements. See also Steps 1b, 1c, 1e, and 1f.
7a	If more than one NCP client is to be in session with NCPROUTE, repeat Step 7 to configure SNALINK LU0 for another session. TCP/IP definitions must be defined for each SNALINK LU0 session. If TCP/IP is currently running and another NCP client is to be added, another SNALINK LU0 can be configured using OBEYFILE commands. This allows TCP/IP to be reconfigured without having to shut down TCP/IP.
8	If you are using OROUTED, make sure that the routing parameters (BSDROUTINGPARMS statement) for the NCP clients are defined correctly. In addition, directly-connected passive routes to the NCP clients must be defined in the .ETC.GATEWAYS data set.
9	If you are using SNMP, make sure that the SNMP agent is configured correctly in the <i>hlq</i> .PROFILE.TCPIP data set. If the SNMP client is on an MVS host, verify that the SNMP client address space is also configured. The cataloged procedure names (SNMPD and SNMPQE) for the SNMP agent and client are referred to on the OBEY, AUTOLOG (optional), and PORT statements.
9a	For the SNMP agent, make sure that the access authority information is defined correctly in the SEZAINST(EZBNRPRF) data set for the NCPROUTE profile, referenced in the NCPROUTE cataloged procedure.

Table 37. Diagnostic steps for NCPROUTE connection problems (continued)

Step	Action
10	If an NCP client is activated and ready to establish a session with NCPROUTE, make sure that the cataloged procedures for TCPIP, NCPROUTE, and SNALINK are all started. If you are using SNMP, make sure that the SNMP agent and client are started.
10a	Make sure that the SNALINK devices are started by the START statement in the <i>h/q.PROFILE.TCPIP</i> data set. The SNALINK devices can also be started by an OBEYFILE START command.
10b	Make sure that VTAM command prompts at the system operator console are replied to; otherwise, a SNALINK session can be in a pending activation state.
10c	Make sure that the NCP client physical and logical lines for the internet interfaces (token ring and Ethernet) are active.
10d	Make sure that NCST lines are active for the SNALINK LU0 sessions.
10e	Make sure that VTAM cross-domain resource managers (CDRMs) are active in the MVS hosts.
11	For network connectivity problems, see Chapter 4, “Diagnosing network connectivity problems” on page 25.
13	For SNMP problems, see Chapter 20, “Diagnosing Simple Network Management Protocol (SNMP) problems” on page 487.
14	For OROUTED problems, see Chapter 24, “Diagnosing OROUTED problems” on page 581.

Analyzing routing failures

Routing problems are usually the result of outages in a network and when there are no alternative routing paths available for recovery. They can also be the result of incorrect configurations in the channel-attached and network-attached routers, as well as incorrect ARP entries, when applicable. PING and Traceroute commands to and from a z/OS host are useful diagnosis aids for problem determination. In this section, unless otherwise specified, the following command terms are used:

PING

Refers to z/OS UNIX oping, TSO PING, and the ping commands used on other platforms.

Traceroute

Refers to z/OS UNIX otracert/traceroute, TSO TRACERTE, and the traceroute commands used on other platforms

NETSTAT ROUTE

Refers to the z/OS UNIX onetstat -r, TSO NETSTAT ROUTE, and the netstat route commands used on other platforms.

NETSTAT GATE

Refers to the z/OS UNIX onetstat -g and TSO NETSTAT GATE commands. This command is available only on the z/OS platform.

NETSTAT ARP

Refers to the z/OS UNIX onetstat -R ALL, TSO NETSTAT ALL, and the netstat arp commands used on other platforms.

NCPROUTE routing failures are reported when a client is unable to get a positive response to a PING or Traceroute command for a remote host where there are NCPs acting as RIP servers along the routing paths.

Documentation

The following documentation should be available for initial diagnosis of routing failures:

- NCPROUTE console log
- TCP/IP console log
- *hlq.PROFILE.TCPIP* data set
- NCP client network definitions data set (NCP generation)
- Output from MODIFY NCPROUTE, TABLES command for a display of internal tables representing a NCP client.
- Outputs from PING and Traceroute commands.

Analysis

Table 38 shows symptoms of PING failures and refers to the steps needed for initial diagnosis of the error.

Table 38. NCPROUTE routing failures

Routing Failure	Analysis Steps
Incorrect response	1, 2, 3, 4, 5, 6, 7, 8
Timeouts	2, 9

Table 39 gives the diagnostic steps referred to in Table 38.

Note: Since an NCP client cannot respond to Traceroute commands, the PING command may be used to diagnose routing failures. However, a Traceroute command can be used to locate a suspect router along the routing path to a remote host beyond the NCP client. In the steps below, the PING command is used for diagnosis.

Table 39. Diagnostic steps for NCPROUTE routing failures

Step	Action
1	Make sure the PING command contains a valid destination IP address for the remote host.
2	Make sure that a 374x communications controller acting as a RIP server involved in the PING transaction is active and is running with a correct level of NCP LOADLIB. Verify that correct network definitions are defined in the NCP generation and that the NCP client is in session with NCPROUTE.
3	If the PING command was issued from a remote host, issue the NETSTAT ROUTE, NETSTAT GATE (if host is z/OS), and NETSTAT ARP commands from there for its routing and ARP table information.
3a	If the host is running with OROUTED, verify the routes and networks defined in the <i>hlq.ETC.GATEWAYS</i> data set. Directly-connected passive routes to the NCP clients must be defined in the <i>hlq.ETC.GATEWAYS</i> data set to ensure NCPROUTE connectivity with the NCP clients.
3b	If there are any problems with the routes and networks, see "Using Netstat command" on page 30.
4	If the host is running with OROUTED, verify the routes and networks defined in the GATEWAYS file in the ETC subdirectory of TCPIP.
4a	For routers or hosts running on platforms other than z/OS, refer to their documentation for more information on correcting routing problems. Also, refer to these documentations for NETSTAT commands to display the routing and ARP tables for problem determination.

Table 39. Diagnostic steps for NCPROUTE routing failures (continued)

Step	Action
5	If there are no problems with the routes or networks, check for broken or poorly connected cables between the client and the remote host. This includes checking the IP interfaces (token ring and Ethernet) on the 374x communications controller.
6	Make sure there is a channel connection between the 374x communications controller and the MVS host. A channel connection can be interrupted by an Automatic Network Shutdown (ANS) situation. ANS can occur when the system operator puts the MVS console into CP mode. In this case, the system operator will need to return to MVS from CP to recover from ANS.
7	For more information about diagnosing network connectivity problems, refer to Chapter 4, "Diagnosing network connectivity problems" on page 25.
8	For more information about diagnosing PING problems, refer to "Using Ping command" on page 28.
9	For more information about diagnosing PING timeouts, refer to "Correcting timeout problems" on page 29.

Incorrect output

Problems with incorrect output are reported when the data sent to the client is in an unexpected form (for example, incorrect TCTIP output, incorrect SNALINK LU0 output, invalid RIP commands, incorrect RIP broadcasting information, incorrect routing-table updates, truncated packets, or incorrect SNMP agent or client output).

Documentation

The following documentation should be available for initial diagnosis of incorrect output:

- NCPROUTE cataloged procedure
- Documentation for NCPROUTE incorrect output
 - NCPROUTE console log
 - NCPROUTE.PROFILE data set
 - NCP client network definitions data set (NCP generation)
 - Output from MODIFY NCPROUTE, TABLES command for a display of internal tables representing a NCP client.
- Documentation for TCPIP incorrect output
 - TCP/IP console log
 - *hlq*.TCPIP. PROFILE data set
 - TCPIP.DATA data set
- Documentation for SNMP agent incorrect output
 - SNMP console logs for SNMP agent and client
 - *hlq*.MIBDESC.DATA data set
 - *hlq*
 - *hlq*.PW.SRC data set
 - NetView log (if SNMP client is on an MVS host)

Analysis

Table 40 shows types of incorrect output and refers to the steps needed for initial diagnosis of the error.

Table 40. NCPROUTE incorrect output

Incorrect output	Analysis steps
TCP/IP incorrect output	1
SNALINK LU0 incorrect output	2

Table 40. NCPROUTE incorrect output (continued)

Incorrect output	Analysis steps
NCPROUTE incorrect output	3
SNMP agent or client incorrect output	4

Table 41 gives the diagnostic steps referred to in Table 40 on page 627.

Table 41. Diagnostic steps for NCPROUTE incorrect output

Step	Action
1	If the TCP/IP console shows a message, refer to <i>z/OS Communications Server: IP Messages Volume 2 (EZB)</i> and follow the directions for system programmer response for the message.
1a	Information in the TCP/IP console log should contain a detailed description of the error.
1b	In the event of TCP/IP loops or hangs, see Chapter 3, “Diagnosing abends, loops, and hangs” on page 21.
3	If the NCPROUTE console shows a message, refer to <i>z/OS Communications Server: IP Messages Volume 2 (EZB)</i> and follow the directions for system programmer response for the message.
4	If the SNMP agent or client console shows a message, refer to <i>z/OS Communications Server: IP Messages Volume 2 (EZB)</i> and follow the directions for system programmer response for the message.
4a	For more information about diagnosing SNMP problems, see Chapter 20, “Diagnosing Simple Network Management Protocol (SNMP) problems” on page 487.

Session outages

Session outages are reported as an unexpected termination of the TCP/IP connection, the SNALINK LU0 task, the NCPROUTE-to-NCP client session, or the NCPROUTE-to-SNMP agent connection. A session that has been disconnected or ended will result in NCPROUTE being returned to the initial state of waiting for Hello PDUs and SNMP requests from an NCP client.

Documentation

The following documentation should be available for initial diagnosis of session outages:

- Documentation for TCP/IP session outage
 - TCP/IP console log
- Documentation for SNALINK LU0 session outage
 - SNALINK LU0 console log
 - VTAM console log
- Documentation for NCPROUTE-to-NCP client session outage
 - NCPROUTE cataloged procedure
 - NCPROUTE console log
 - NCP client network definitions data set (NCP generation)
- Documentation for NCPROUTE-to-SNMP agent session outage
 - SNMP console log for SNMP agent
 - NetView log (if the SNMP client is on the MVS host)

Analysis

Table 42 on page 629 shows symptoms of session outages and refers to the steps needed for initial diagnosis of the error.

Table 42. NCPROUTE session outages

Session outage	Analysis steps
TCP/IP session outage	1
SNALINK LU0 session outage	2
NCPROUTE-to-NCP client session outage	3
NCPROUTE-to-SNMP agent session outage	4

Table 43 gives the diagnostic steps referred to in Table 42.

Table 43. Diagnostic steps for NCPROUTE session outages

Step	Action
1	If the TCP/IP console shows a TCP/IP error message, refer to <i>z/OS Communications Server: IP Messages Volume 2 (EZB)</i> and follow the directions for system programmer response for the message.
1b	If TCP/IP abended, see Chapter 3, “Diagnosing abends, loops, and hangs” on page 21.
3	If the NCPROUTE console shows an NCPROUTE error message, refer to <i>z/OS Communications Server: IP Messages Volume 2 (EZB)</i> and follow the directions for system programmer response for the message.
4	If the SNMP agent console shows a SNMP error message, refer to <i>z/OS Communications Server: IP Messages Volume 2 (EZB)</i> and follow the directions for system programmer response for the message.
4a	For more information about diagnosing SNMP problems, see Chapter 20, “Diagnosing Simple Network Management Protocol (SNMP) problems” on page 487.

NCPROUTE traces

There are many TCP/IP traces that can be useful in identifying the cause of NCPROUTE problems. This section discusses the NCPROUTE traces.

Note: NCPROUTE trace output is sent to the location specified by the SYSPRINT DD statement in the NCPROUTE cataloged procedure.

Activating NCPROUTE global traces

The NCPROUTE global traces are all controlled by parameters on PARMs= in the PROC statement of the NCPROUTE cataloged procedure. (*Global tracing* means that all NCP clients are traced.)

For example:

```
//NCPROUT PROC MODULE=NCPROUTE,PARMS='/-t -t'
```

Note: These parameters are also valid when starting the NCPROUTE server with the START command.

The NCPROUTE parameters that control global tracing are:

- t** Activates global tracing of actions for all NCP clients.
- t -t** Activates global tracing of packets for all NCP clients. NCPROUTE tracing

can be started and stopped using the MODIFY command. For more information, refer to the *z/OS Communications Server: IP Configuration Reference*.

- tq** Deactivates tracing at all levels. This parameter suppresses tracing for all NCP clients and overrides the trace settings on the GATEWAY statements in the NCPROUTE GATEWAYS data set.
- dp** Activates global tracing of data packets coming in and out of NCPROUTE. The data is displayed in data format.

Notes:

1. A slash (/) must precede the first parameter.
2. Each parameter must be separated by a blank.
3. Mixed case is allowed for the parameters.
4. The parameters for the NCPROUTE procedure are case-sensitive.
5. There are no third- or fourth-level global tracing options like those on the GATEWAY statements in the NCPROUTE GATEWAYS data set. The system uses the higher of the two settings for a specific NCP client.
6. The data packets trace option is not available for selective tracing.
7. The parameters described here include only those that activate tracing. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about all of the NCPROUTE parameters.

Activating NCPROUTE selective traces

The NCPROUTE selective traces are all activated as trace options specified in the OPTIONS statement for an NCP client in the NCPROUTE GATEWAYS data set. Selective tracing means a different trace level can be specified for each NCP client. To assist in problem isolation, a particular NCP client can be selected for tracing.

The keyword on the OPTIONS statement that controls selective tracing for an NCP client is `trace.level`. The value that follows this keyword indicates the trace level to be used.

Value	Meaning
0	Does not activate any traces.
1	Activates tracing of actions by the NCPROUTE server.
2	Activates tracing of all packets sent or received.
3	Activates tracing of actions, packets sent or received, and packet history. Circular trace buffers are used for each interface to record the history of all packets traced. This history is included in the trace output whenever an interface becomes inactive.
4	Activates tracing of actions, packets sent or received, packet history, and packet contents. The RIP network routing information is included in the trace output.

Notes:

1. The selective traces must be defined prior to activation of an NCP client or prior to starting the NCPROUTE cataloged procedure.
2. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about the GATEWAYS data set and the GATEWAY and OPTIONS statements.

For example, the following command would activate tracing of actions, packets sent or received, packet history, and packet contents:

```
options trace.level 4
```

NCPROUTE trace example and explanation

Figure 71 on page 632 shows an example of an NCPROUTE trace with actions, packets, history, and contents traced. The trace was generated with trace level 4 specified in the `OPTIONS` statement and `PARMS='/ -t -t -dp'` in the `PROC` statement of the NCPROUTE cataloged procedure.

The trace level column does not appear in the actual trace. It was added to the example to indicate the levels of the trace for which the line will be generated. For example, including: `trace.level 3` on the options statement NCP client GATEWAYS data set would result in a level 3 trace, and all of the lines indicated as trace level **1**, **2**, or **3** would be generated in the trace output. Lines indicated as trace level **d** are generated if the `-dp` parameter is specified.

```

Trace
level
0 1 15:29:48 EZB3826I Port 580 assigned to ncprout
0 15:29:49 EZB3885I Input parameter(s): -t -t -dp
1 15:29:49 EZB4159I Global tracing actions started
2 15:29:49 EZB4160I Global tracing packets started
0 15:29:49 EZB3834I *****
0 2 15:29:49 EZB4196I * Opening NCPROUTE profile dataset (DD:NCPRPROF)
0 15:29:49 EZB3834I *****
0 3 15:29:49 EZB4055I ** Attempting to (re)start SNMP connection
0 15:29:49 EZB4059I Connecting to agent 9.67.116.66 on DPI port 1141
0 15:29:49 EZB4062I SNMP DPI connection established
0 15:29:50 EZB4064I 1.3.6.1.4.1.2.6.17. registered with SNMP agent0
1 15:29:50 EZB3829I Waiting for incoming packets
d 4 ===== Received datagram from NCP client (length=32)
d 0000 0100 0000 c1f0 f4d5
d 0008 f7f1 f1d7 f0f5 61f0
d 0010 f661 f9f4 40f1 f07a
d 0018 flf0 7af4 f200 0000
d 0020(32)
0 15:29:51 EZB3834I *****
0 15:29:51 EZB3876I * Hello from new client 9.67.116.65
0 15:29:51 EZB3877I * RIT dataset name: A04N711P
0 15:29:51 EZB3878I * RIT ID: 05/06/94 10:10:42
0 15:29:51 EZB3834I *****
0 15:29:51 EZB3867I Acknowledge to 9.67.116.65: Hello Received
0 15:29:51 EZB3999I Establishing session with client 9.67.116.65
0 15:29:51 EZB3868I Acknowledge to 9.67.116.65: RIT Loaded OK
0 15:29:51 EZB4166I Session with client 9.67.116.65 started
1 15:29:51 EZB3829I Waiting for incoming packets
1 ===== Received datagram from NCP client (length=8)
d 0000 0800 0000 0a44 005c
d 0008(8)
0 15:29:51 EZB3834I *****
0 5 15:29:51 EZB3898I * Recv: Inactive Interface List from 9.67.116.65
0 * 1 interface(s) found:
0 15:29:51 EZB3899I * 10.68.0.92 - TR92
0 15:29:51 EZB3834I *****
0 15:29:51 EZB3834I *****
0 6 15:29:51 EZB3956I * Processing interface NCSTALU1
0 15:29:51 EZB3834I *****
0 15:29:51 EZB3959I Point-to-point interface, using dstaddr
0 15:29:51 EZB3962I Adding (sub)network address for interface
1 15:29:51 EZB3912I ifwithnet: compare with NCSTALU1
1 15:29:51 EZB3915I netmatch 9.67.116.65 and 9.67.116.65
1 15:29:51 EZB4029I Tue Jun 28 15:29:51:
1 7 15:29:52 EZB4030I ADD destination 9.67.116.66, router 9.67.116.66, metric 1
1 flags UP|HOST state INTERFACE|CHANGED|INTERNAL|PERM|SUBNET timer 0

```

Figure 71. NCPROUTE trace (Part 1 of 10)

```

0      15:29:52 EZB3834I *****
0      15:29:52 EZB3956I * Processing interface TR88
0      15:29:52 EZB3834I *****
0      15:29:52 EZB3960I This interface is not point-to-point
0      15:29:52 EZB3962I Adding (sub)network address for interface
1      15:29:52 EZB3912I ifwithnet: compare with NCSTALU1
1      15:29:52 EZB3912I ifwithnet: compare with TR88
1      15:29:52 EZB3915I netmatch 10.68.0.88 and 10.68.0.88
1      15:29:52 EZB4030I ADD destination 10.0.0.0, router 10.68.0.88, metric 1
1                      flags UP state INTERFACE|CHANGED|INTERNAL|SUBNET|PERM timer 0
1      15:29:52 EZB3912I ifwithnet: compare with NCSTALU1
1      15:29:52 EZB3912I ifwithnet: compare with TR88
1      15:29:52 EZB3915I netmatch 10.68.0.88 and 10.68.0.88
1      15:29:52 EZB4030I ADD destination 10.68.0.0, router 10.68.0.0, metric 1
1                      flags UP state INTERFACE|CHANGED|SUBNET|PERM timer 0
0      15:29:52 EZB3834I *****
0      15:29:52 EZB3956I * Processing interface TR92
0      15:29:52 EZB3834I *****
0      15:29:52 EZB3960I This interface is not point-to-point
0      15:29:52 EZB3948I Interface TR92 not up
0      15:29:52 EZB3834I *****
0      8 15:29:52 EZB3973I * Opening GATEWAYS dataset for client 9.67.116.65
0                      * 'TCPCS.NCPRROUTE.GATEWAYS(A04N711P)'
0      15:29:52 EZB3834I *****
0      15:29:52 EZB3968I Start of GATEWAYS processing:
0      15:29:52 EZB4195I Option(s): trace.level 4 supply on default.router no
1      15:29:52 EZB4015I Client tracing actions started
2      15:29:52 EZB4016I Client tracing packets started
3      15:29:52 EZB4017I Client tracing history started
4      15:29:52 EZB4018I Client tracing packet contents started
0      15:29:52 EZB4198I (no etc.gateway definitions)
0      15:29:52 EZB4150I End of GATEWAYS processing
1      15:29:52 EZB3829I Waiting for incoming packets
d      9 ===== Received datagram from NCP client (length=80)
d      10 0000 0700 0000 9200 004a
0008 4500 0048 09c0 0000
d      0010 3c11 79dc 0943 7442
d      0018 0943 7441 0208 0208
d      0020 0034 079e 0201 0000
d      0028 0002 0000 0943 7441
d      0030 0000 0000 0000 0000
d      0038 0000 0001 0002 0000
d      0040 0943 7000 0000 0000
d      0048 0000 0000 0000 0001
d      0050(80)
d      ===== Transport PDU header (length=8)
d      0000 0700 0000 9200 004a
d      0008(8)
d      ===== IP header (length=20)
d      0000 4500 0048 09c0 0000
d      0008 3c11 79dc 0943 7442
d      0010 0943 7441 8002 c12c
d      0018(24)

```

Figure 71. NCPRROUTE trace (Part 2 of 10)

```

d      ===== UDP header (length=8)
d      0000      0208 0208 0034 079e
d      0008(8)
d      ===== UDP data (length=44)
d      0000      0201 0000 0002 0000
d      0008      0943 7441 0000 0000
d      0010      0000 0000 0000 0001
d      0018      0002 0000 0943 7000
d      0020      0000 0000 0000 0000
d      0028      0000 0001 0001 6e68
d      0030(48)
1 11 15:30:04 EZB3894I Transport from 9.67.116.65: 44 bytes of RIP data
2 15:30:04 EZB4045I      RESPONSE from 9.67.116.66 -> 520:
d 12 ===== RIP net info (length=20)
d      0000      0002 0000 0943 7441
d      0008      0000 0000 0000 0000
d      0010      0000 0001 8002 c12c
d      0018(24)
4 15:30:04 EZB4049I      destination 9.67.116.65 metric 1
d      ===== RIP net info (length=20)
d      0000      0002 0000 0943 7000
d      0008      0000 0000 0000 0000
d      0010      0000 0001 8002 c12c
4 0018(24)
1 15:30:04 EZB4049I      destination 9.67.112.0 metric 1
1 15:30:04 EZB4029I Tue Jun 28 15:30:04:
1 13 15:30:04 EZB4030I ADD destination 9.67.112.0, router 9.67.116.66, metric 2
1      flags UP|GATEWAY state CHANGED|SUBNET timer 0
1 14 15:30:04 EZB3855I NCP_Add out to 9.67.116.65
1      Route to: 9.67.112.0 via interface 9.67.116.65 to 9.67.116.66
1      Metric: 2, Type Subnet
1 15:30:04 EZB3829I Waiting for incoming packets
1 15 15:30:20 EZB4011I client 9.67.116.65: 30 second timer expired (broadcast)
1 15:30:20 EZB3951I client 9.67.116.65: supply 9.67.116.66 -> 0 via NCSTALU1
4 16 15:30:20 EZB4045I      RESPONSE to 9.67.116.66 -> 0:
d      ===== RIP net info (length=20)
d      0000      0002 0000 0943 7442
d      0008      0000 0000 0000 0000
d      0010      0000 0001 8002 c12c
d      0018(24)
4 15:30:20 EZB4049I      destination 9.67.116.66 metric 1
d      ===== RIP net info (length=20)
d      0000      0002 0000 0a00 0000
d      0008      0000 0000 0000 0000
d      0010      0000 0001 8002 c12c
d      0018(24)
4 15:30:20 EZB4049I      destination 10.0.0.0 metric 1
d      ===== RIP net info (length=20)
d      0000      0002 0000 0943 7000
d      0008      0000 0000 0000 0000
d      0010      0000 0002 8002 c12c
d      0018(24)
4 15:30:20 EZB4049I      destination 9.67.112.0 metric 2

```

Figure 71. NCPROUTE trace (Part 3 of 10)

```

d      ===== UDP data (length=64)
d      0000      0201 0000 0002 0000
d      0008      0943 7442 0000 0000
d      0010      0000 0000 0000 0001
d      0018      0002 0000 0a00 0000
d      0020      0000 0000 0000 0000
d      0028      0000 0001 0002 0000
d      0030      0943 7000 0000 0000
d      0038      0000 0000 0000 0002
d      0040(64)
d      ===== UDP header (length=8)
d      0000      0208 0208 0048 fd70
d      0008(8)
d      ===== IP header (length=20)
d      0000      4500 005c 0000 0000
d      0008      0411 bb88 0943 7441
d      0010      0943 7442 8002 c12c
d      0018(24)
d      ===== Transport PDU header (length=8)
d      0000      0700 0000 0943 7441
d      0008(8)
d      ===== Sending Transport PDU to NCP client (length=100)
d      0000      0700 0000 0943 7441
d      0008      4500 005c 0000 0000
d      0010      0411 bb88 0943 7441
d      0018      0943 7442 0208 0208
d      0020      0048 fd70 0201 0000
d      0028      0002 0000 0943 7442
d      0030      0000 0000 0000 0000
d      0038      0000 0001 0002 0000
d      0040      0a00 0000 0000 0000
d      0048      0000 0000 0000 0001
d      0050      0002 0000 0943 7000
d      0058      0000 0000 0000 0000
d      0060      0000 0002 0000 0001
d      0068(104)
d      .
d      .
1      17 15:30:20 EZB3948I Interface TR92 not up
1      15:30:20 EZB3829I Waiting for incoming packets
d      .
1      15:30:20 EZB3894I Transport from 9.67.116.65: 64 bytes of RIP data
2      15:30:20 EZB4045I      RESPONSE from 10.68.0.88 -> 520:
d      .
4      15:30:20 EZB4049I      destination 9.67.116.66 metric 1
d      .
4      15:30:20 EZB4049I      destination 10.68.0.0 metric 1
d      .
4      15:30:20 EZB4049I      destination 9.67.112.0 metric 2
1      15:30:20 EZB3829I Waiting for incoming packets
d      .
d      .
d      .

```

Figure 71. NCPROUTE trace (Part 4 of 10)

```

1      15:30:34 EZB3829I Waiting for incoming packets
1      15:30:50 EZB4011I client 9.67.116.65: 30 second timer expired (broadcast)
1      15:30:50 EZB3951I client 9.67.116.65: supply 9.67.116.66 -> 0 via NCSTALU1
2      15:30:50 EZB4045I      RESPONSE to 9.67.116.66 -> 0:
.
4      15:30:50 EZB4049I      destination 9.67.116.66 metric 1
.
4      15:30:50 EZB4049I      destination 10.0.0.0 metric 1
.
4      15:30:50 EZB4049I      destination 9.67.112.0 metric 2
.
1      15:30:50 EZB3951I client 9.67.116.65: supply 10.68.15.255 -> 0 via TR88
2      15:30:50 EZB4045I      RESPONSE to 10.68.15.255 -> 0:
.
4      15:30:50 EZB4049I      destination 9.67.116.66 metric 1
.
4      15:30:50 EZB4049I      destination 10.68.0.0 metric 1
.
4      15:30:50 EZB4049I      destination 9.67.112.0 metric 2
.
.
1      15:32:35 EZB3894I Transport from 9.67.116.65: 64 bytes of RIP data
2      15:32:35 EZB4045I      RESPONSE from 9.67.116.66 -> 520:
.
4      15:32:35 EZB4049I      destination 9.67.116.65 metric 1
.
4      15:32:35 EZB4049I      destination 10.0.0.0 metric 2
.
4      15:32:35 EZB4049I      destination 9.67.112.0 metric 16
1      15:32:35 EZB4029I Tue Jun 28 15:32:35:
1      18 15:32:35 EZB4036I CHANGE metric destination 9.67.112.0, router 9.67.116.66, from 2 to 16
19 15:32:35 EZB3862I NCP_Delete out to 9.67.116.65:
      Route to 9.67.112.0, type = Subnet
1      15:32:35 EZB3943I Send dynamic update
1      15:32:35 EZB3950I toall: requested to skip interface NCSTALU1
1      15:32:35 EZB3951I client 9.67.116.65: supply 10.68.15.255 -> 0 via TR88
2      15:32:35 EZB4045I      RESPONSE to 10.68.15.255 -> 0:
.
4      15:32:35 EZB4049I      destination 9.67.112.0 metric 16
.
1      15:32:35 EZB3948I Interface TR92 not up
1      15:32:35 EZB3945I Inhibit dynamic update for 2017537 usec
1      15:32:35 EZB3829I Waiting for incoming packets
.
1      15:32:35 EZB3894I Transport from 9.67.116.65: 24 bytes of RIP data
1      15:32:35 EZB4045I      RESPONSE from 10.68.0.88 -> 520:
.
4      15:32:35 EZB4049I      destination 9.67.112.0 metric 16
1      15:32:35 EZB3829I Waiting for incoming packets

```

Figure 71. NCPROUTE trace (Part 5 of 10)

```

1      15:32:50 EZB4011I client 9.67.116.65: 30 second timer expired (broadcast)
1      15:32:50 EZB3951I client 9.67.116.65: supply 9.67.116.66 -> 0 via NCSTALU1
2      15:32:50 EZB4045I      RESPONSE to 9.67.116.66 -> 0:
      .
4      15:32:50 EZB4049I      destination 9.67.116.66 metric 1
      .
4      15:32:50 EZB4049I      destination 10.0.0.0 metric 1
      .
4      15:32:50 EZB4049I      destination 9.67.112.0 metric 16
1      15:32:50 EZB3951I client 9.67.116.65: supply 10.68.15.255 -> 0 via TR88
2      15:32:50 EZB4045I      RESPONSE to 10.68.15.255 -> 0:
      .
1      15:32:50 EZB3948I Interface TR92 not up
1      15:32:50 EZB3829I Waiting for incoming packets
:
1      15:36:15 EZB3829I Waiting for incoming packets
1      20 15:36:39 EZB4009I client 9.67.116.65: 5 minute timer expired for route to 9.67.112.0
1      15:36:39 EZB4029I Tue Jun 28 15:36:39:
1      21 15:36:39 EZB4030I DELETE destination 9.67.112.0, router 9.67.116.66, metric 16
1      flags UP|GATEWAY state SUBNET timer 300
1      15:36:39 EZB4011I client 9.67.116.65: 30 second timer expired (broadcast)
1      15:36:39 EZB3951I client 9.67.116.65: supply 9.67.116.66 -> 0 via NCSTALU1
2      15:36:39 EZB4045I      RESPONSE to 9.67.116.66 -> 0:
      .
4      15:36:39 EZB4049I      destination 9.67.116.66 metric 1
      .
4      15:36:39 EZB4049I      destination 10.0.0.0 metric 1
      .
1      15:36:39 EZB3951I client 9.67.116.65: supply 10.68.15.255 -> 0 via TR88
2      15:36:39 EZB4045I      RESPONSE to 10.68.15.255 -> 0:
      .
4      15:36:39 EZB4049I      destination 9.67.116.66 metric 1
      .
4      15:36:39 EZB4049I      destination 10.68.0.0 metric 1
      .
:
1      22 15:43:01 EZB3895I Transport from 9.67.116.65: 43 bytes of SNMP data
1      23 15:43:01 EZB4182I SNMP request received from NCP client 9.67.116.65
d      ===== Object data (length=13)
d      0000      2b06 0102 0104 1501
d      0008      0709 4374 4207 39f8
d      0010(16)
d      ===== prefix + address (length=12)
d      0000      2b06 0104 0102 0611
d      0008      0943 7441 4207 39f8
d      0010(16)

```

Figure 71. NCPROUTE trace (Part 6 of 10)

```

d      ===== Inbound SNMP packet (post edit) (length=55)
d      0000 3035 0201 0004 0473
d      0008 6e6d 70a0 2a02 0115
d      0010 0201 0002 0100 301f
d      0018 301d 0619 2b06 0104
d      0020 0102 0611 0943 7441
d      0028 2b06 0102 0104 1501
d      0030 0709 4374 4205 0000
d      0038(56)
d      ===== Sending SNMP request to agent (length=55)
d      0000 3035 0201 0004 0473
d      0008 6e6d 70a0 2a02 0115
d      0010 0201 0002 0100 301f
d      0018 301d 0619 2b06 0104
d      0020 0102 0611 0943 7441
d      0028 2b06 0102 0104 1501
d      0030 0709 4374 4205 00f3
d      0038(56)
1      15:43:01 EZB3829I Waiting for incoming packets
1      15:43:01 EZB4194I SNMP sub-agent received DPI request
d      ===== Received DPI request from SNMP agent (length=69)
d      0000 0043 0201 0101 f14b
d      0008 f34b f64b f14b f44b
d      0010 f14b f24b f64b f1f7
d      0018 4bf9 4bf6 f74b f1f1
d      0020 f64b f6f5 4bf4 f34b
d      0028 f64b f14b f24b f14b
d      0030 f44b f2f1 4bf1 4bf7
d      0038 4bf9 4bf6 f74b f1f1
d      0040 f64b f6f6 0007 2b30
d      0048(72)
1      15:43:01 EZB4072I SNMP sub-agent:DPI GET request
                        (1.3.6.1.4.1.2.6.17.9.67.116.65.43.6.1.2.1.4.21.1.7.9.67.116.66) received
1      15:43:01 EZB4083I iproutenexthop.9.67.116.66
d      ===== Sending DPI response to SNMP agent (length=77)
d      0000 004b 0201 0105 00f1
d      0008 4bf3 4bf6 4bf1 4bf4
d      0010 4bf1 4bf2 4bf6 4bf1
d      0018 f74b f94b f6f7 4bf1
d      0020 f1f6 4bf6 f54b f4f3
d      0028 4bf6 4bf1 4bf2 4bf1
d      0030 4bf4 4bf2 f14b f14b
d      0038 f74b f94b f6f7 4bf1
d      0040 f1f6 4bf6 f600 8500
d      0048 0409 4374 4149 5f3c
d      0050(80)
1      15:43:01 EZB3829I Waiting for incoming packets
1      15:43:01 EZB4068I SNMP response received from agent 9.67.116.66

```

Figure 71. NCPROUTE trace (Part 7 of 10)


```

d      ===== Received SNMP response from agent (length=59)
d      0000      3039 0201 0004 0473
d      0008      6e6d 70a2 2e02 0115
d      0010      0201 0002 0100 3023
d      0018      3021 0619 2b06 0104
d      0020      0102 0611 0943 7441
d      0028      2b06 0102 0104 1501
d      0030      0709 4374 4240 0409
d      0038      4374 4196 95a2 8540
d      0040(64)
d      ===== Object data (length=25)
d      0000      2b06 0104 0102 0611
d      0008      0943 7441 2b06 0102
d      0010      0104 1501 0709 4374
d      0018      4240 2910 0000 0001
d      0020(32)
d      ===== prefix + address (length=12)
d      0000      2b06 0104 0102 0611
d      0008      0943 7441 2b06 0102
d      0010(16)
d      ===== Outbound SNMP packet (post edit) (length=47)
d      0000      302d 0201 0004 0473
d      0008      6e6d 70a2 2202 0115
d      0010      0201 0002 0100 3017
d      0018      3015 060d 2b06 0102
d      0020      0104 1501 0709 4374
d      0028      4240 0409 4374 4100
d      0030(48)
1      15:43:01 EZB4172I SNMP reply sent to NCP client 9.67.116.66
d      ===== UDP data (length=47)
d      0000      302d 0201 0004 0473
d      0008      6e6d 70a2 2202 0115
d      0010      0201 0002 0100 3017
d      0018      3015 060d 2b06 0102
d      0020      0104 1501 0709 4374
d      0028      4240 0409 4374 4168
d      0030(48)
d      ===== UDP header (length=8)
d      0000      00a1 040e 0037 ec9f
d      0008(8)
d      ===== IP header (length=20)
d      0000      4500 004b 0034 0000
d      0008      0411 a18e 0a44 0058
d      0010      0a44 0001 8002 c12c
d      0018(24)
d      ===== Transport PDU header (length=8)
d      0000      0700 0000 0a44 0058
d      0008(8)

```

Figure 71. NCPROUTE trace (Part 8 of 10)

```

d      ===== Sending Transport PDU to NCP client (length=84)
d      0000      0700 0000 0a44 0058
d      0008      4500 004b 0034 0000
d      0010      0411 a18e 0a44 0058
d      0018      0a44 0001 00a1 040e
d      0020      0037 ec9f 302d 0201
d      0028      0004 0473 6e6d 70a2
d      0030      2202 0115 0201 0002
d      0038      0100 3017 3015 060d
d      0040      2b06 0102 0104 1501
d      0048      0709 4374 4240 0409
d      0050      4374 4100 0007 3568
d      0058(88)
1      15:43:01 EZB3829I Waiting for incoming packets
:
0      15:44:30 EZB3834I *****
0      24 15:44:30 EZB3890I * Recv: status from 9.67.116.65
0      15:44:30 EZB3891I * Interface: 10.68.0.88 is now inactive - TR88
0      15:44:30 EZB3834I *****
3      25 15:44:30 EZB4038I *** Packet history for interface TR88 ***
3      15:44:30 EZB4044I Output: trace:
3      15:44:30 EZB4045I     RESPONSE to 10.68.15.255 -> 0:
3      15:44:30 EZB4049I     . destination 9.67.116.66 metric 1
3      15:44:30 EZB4049I     . destination 10.68.0.0 metric 1
3      15:44:30 EZB4049I     . destination 9.67.112.0 metric 2
3      15:44:30 EZB4045I     RESPONSE to 10.68.15.255 -> 0:
3      15:44:30 EZB4049I     . destination 9.67.116.66 metric 1
3      15:44:30 EZB4049I     . destination 10.68.0.0 metric 1
3      15:44:30 EZB4049I     . destination 9.67.112.0 metric 2
3      15:44:30 EZB4045I     RESPONSE to 10.68.15.255 -> 0:
3      15:44:30 EZB4049I     . destination 9.67.116.66 metric 1
3      15:44:30 EZB4049I     . destination 10.68.0.0 metric 1
3      15:44:30 EZB4044I Input: trace:
3      15:44:30 EZB4045I     RESPONSE from 10.68.0.88 -> 520:
3      15:44:30 EZB4049I     . destination 9.67.116.66 metric 1
3      15:44:30 EZB4049I     . destination 10.68.0.0 metric 1
3      15:44:30 EZB4049I     . destination 9.67.112.0 metric 2

```

Figure 71. NCPROUTE trace (Part 9 of 10)

```

3      15:44:30 EZB4045I      RESPONSE from 10.68.0.88 -> 520:
3      15:44:30 EZB4049I      .
3      15:44:30 EZB4049I      destination 9.67.116.66 metric 1
3      15:44:30 EZB4049I      .
3      15:44:30 EZB4049I      destination 10.68.0.0 metric 1
3      15:44:30 EZB4049I      .
3      15:44:30 EZB4049I      destination 9.67.112.0 metric 2
3      15:44:30 EZB4045I      RESPONSE from 10.68.0.88 -> 520:
3      15:44:30 EZB4049I      .
3      15:44:30 EZB4049I      destination 9.67.116.66 metric 1
3      15:44:30 EZB4049I      .
3      15:44:30 EZB4049I      destination 10.68.0.0 metric 1
3      15:44:30 EZB4049I      .
3      15:44:30 EZB4049I      destination 9.67.112.0 metric 2
3      15:44:30 EZB4045I      RESPONSE from 10.68.0.88 -> 520:
3      15:44:30 EZB4049I      .
3      15:44:30 EZB4049I      destination 9.67.116.66 metric 1
3      15:44:30 EZB4049I      .
3      15:44:30 EZB4049I      destination 10.68.0.0 metric 1
3      15:44:30 EZB4039I *** End packet history ***
3      15:44:31 EZB3829I Waiting for incoming packets
3      15:44:31 EZB3829I .
3      15:44:31 EZB3829I .
3      15:44:31 EZB3829I .
1      15:44:41 EZB3948I Interface TR88 not up
1      15:44:41 EZB3948I Interface TR92 not up
1      15:44:41 EZB3829I Waiting for incoming packets
1      15:44:41 EZB3829I .
1      15:44:41 EZB3829I .
1      15:44:41 EZB3829I .

```

Figure 71. NCPROUTE trace (Part 10 of 10)

The following information explains the numbered items in the trace:

- 1** The port number and the service name are defined as 580 and ncprout in the *hlq.ETC.SERVICES* data set for this NCPROUTE server.
- 2** NCPROUTE is processing the NCPROUTE.PROFILE definitions.
- 3** NCPROUTE is establishing the connection with the SNMP agent defined in NCPROUTE.PROFILE.
- 4** The NCP client is starting the hand-shaking process with NCPROUTE. NCPROUTE is establishing a session with the NCP client.
- 5** NCPROUTE received a list of inactive interfaces from the NCP client.
- 6** NCPROUTE is initializing its interface tables with interface information from the NCP client.
- 7** NCPROUTE is adding a route to its interface tables.
- 8** NCPROUTE is processing the NCP client GATEWAYS data set. The trace shows NCPROUTE server options and no additional gateway definitions.
- 9** NCPROUTE received a transport datagram from the NCP client.
- 10** The trace shows the contents of the datagram in hexadecimal followed by a division of the datagram into its parts (transport PDU header, IP header, UDP header, and UDP data).
- 11** The trace shows that the NCP client 9.67.116.65 received the broadcasted routing tables from adjacent router 9.67.116.66.

- 12** The UDP data in the datagram contains two routing table entries.
- 13** NCPROUTE is adding a new route to its tables from the information received in the transport datagram.
- 14** NCPROUTE is issuing a request to the NCP client to add the route to its tables.
- 15** The NCP client 30-second timer has expired, so NCPROUTE will supply its routing tables to other routers.
- 16** NCPROUTE is responding to the request by sending its routing tables to the requesting router for the NCP client.
- 17** This line shows an inactive state for interface TR92.
- 18** The NCP client 3-minute timer expired. The client was broadcast a network unreachable route (in the range metric 16—infinite), so NCPROUTE will update its routing tables for the NCP client.
- 19** NCPROUTE is deleting the NCP client from its tables.
- 20** The NCP client five-minute timer has expired for the route to 9.67.112.0.
- 21** NCPROUTE is deleting the route to 9.67.112.0 from its tables for the NCP client.
- 22** NCPR received a transport datagram from the SNMP client through NCP client 9.67.116.65.
- 23** NCPROUTE is processing the SNMP request.
- 24** NCPROUTE has received a status notification from the NCP client. The interface TR88 has become inactive.
- 25** The packet history for the interface TR88 is included in the trace because the interface has become inactive.

Chapter 27. Diagnosing X.25 NPSI problems

The X.25 NPSI server uses an X.25 network or point-to-point X.25 line to transfer TCP/IP traffic. The X.25 NPSI server is a VTAM application running as a started task. Either the NPSI Generalized Access to X.25 Transport Extension (GATE) or Dedicated Access to X.25 Transport Extension (DATE) can be used. GATE is recommended because it allows NPSI to handle more details of error recovery and allows an X.25 physical link to be shared with other functions.

Details of the GATE and DATE programming interfaces are in *X.25 NPSI Host Programming*, and further diagnostic information is in *X.25 NPSI Diagnosis, Customization, and Tuning*. Specifications for carriage of IP traffic on X.25 networks can be found in:

RFC 877

A Standard for the Transmission of IP Datagrams Over Public Data Networks

X25.DOC

Old DDN X.25 specifications from BBN (available by anonymous FTP from nic.ddn.mil in directory netinfo)

RFC 1236

IP to X.121 Address Mapping for DDN

RFC 1356

Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode

Figure 72 on page 644 shows the X.25 NPSI environment.

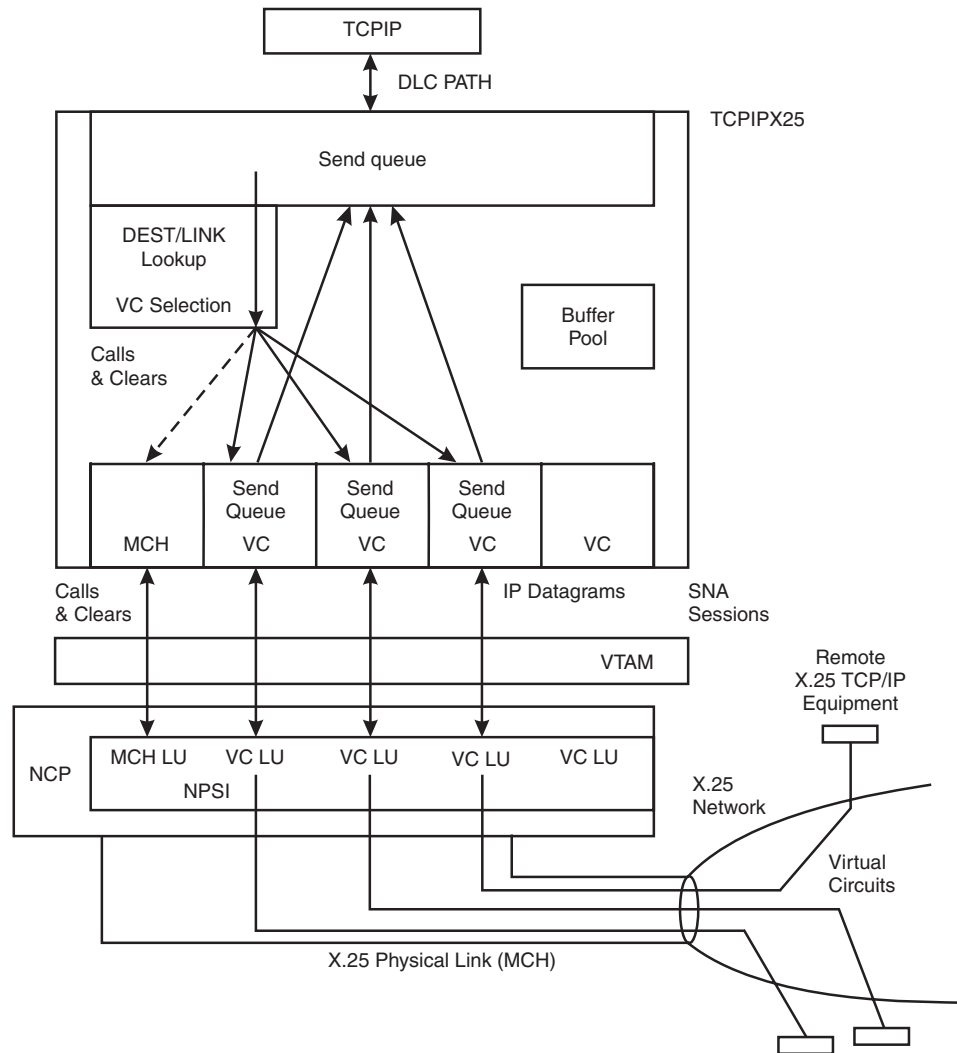


Figure 72. X.25 NPSI environment

Operation

The X.25 NPSI server uses NPSI to set up X.25 virtual circuits as needed to carry traffic to and from remote X.25 equipment. The three main functional areas shown in Figure 72 are:

- TCP/IP interface
- NPSI interface
- IP/X.25 address mapping

IP datagrams are transferred between TCPIP and the X.25 NPSI server on an DLC path established when a TCPIP X25NPSI device is started. The transfer protocol is similar to that used with SNALINK, with the addition of a first-hop IP address passed by TCPIP from the relevant GATEWAY entry. The X.25 NPSI server uses the first hop IP address to look up an X.25 address in its destination table.

Communication with NPSI is by way of several SNA sessions. One control session is established at initialization for each MCH LU defined in a LINK statement in the X.25 NPSI server configuration data set. Commands to establish and terminate X.25 virtual circuit connections pass between the X.25 NPSI server and NPSI on the control session. Refer to *X.25 NPSI Host Programming* for details of the control commands. As new virtual circuits are established, NPSI initiates new SNA sessions with the X.25 NPSI server application by means of VTAM LOGON. IP datagrams are then exchanged with the remote equipment over the VC session until an idle timeout occurs or the VC is taken for another destination.

IP addresses are mapped to X.25 addresses by table lookup, or in the case of the DDN network, by a calculation described in RFC 1236. The X.25 NPSI server performs the lookup with the first-hop IP address on each datagram it receives from TCPIP. The LINK and DEST entries defined in the X.25 NPSI server configuration data set are scanned in order from top to bottom to find a DEST with a matching IP address. Once the DEST is found, the link it applies to is selected to carry the datagram, and the active virtual circuits on that link are scanned to find one with an X.25 address which matches the DEST. If such a VC is found, the datagram is queued for transmission on that VC; if none is found and there is a free VC, a new X.25 call is initiated; if all VCs on the link are in use, the least recently used connection is cleared, as long as it has been open for at least the minimum open time, and a new call is initiated. If no VC matches these conditions, the datagram is discarded.

Configuration requirements

The next two sections describe configuration considerations.

RACF/Security Manager requirement

The user ID assigned to the X.25 NPSI start procedure needs an OMVS Segment assigned to it.

VTAM considerations

- APPL definition
The X.25 NPSI server requires AUTH=(ACQ) and PARSESS=YES in the VTAM APPL definition.
- SWNET definition for switched circuits
 - The value specified for MAXDATA for the PU must be at least 10 bytes greater than the value specified for the maximum packet size on the BUFFERS statement in the X.25 NPSI server configuration data set.
 - SSCPFM=USSNTO and DISCNT=(YES,F) are necessary.

NPSI considerations

- BUILD definition
The value specified for X25.MAXPIU must be at least 10 bytes greater than the value specified for the maximum packet size on the BUFFERS statement in the X.25 NPSI server configuration data set.
- X25.MCH definition
 - LOGAPPL can be coded for recovery.
 - TRAN=NO is required with GATE=DEDICAT.
- X25.VC definition
 - Permanent virtual circuits (PVCs) are not supported.

- Do not code LOGAPPL except with CONNECT=YES (Fast connect).
- Do not code MAXDATA except with CONNECT=YES (Fast connect).
- X25.OUFT definition
 - X.25 facilities specified with X25.OUFT are not used by the X.25 NPSI server.

Sources of diagnostic information

Many problems with the X.25 NPSI server are the result of configuration faults. Configuration files to check are:

- DEVICE, LINK, and GATEWAY entries in PROFILE.TCPIP
- The X.25 NPSI server configuration data set
- VTAM APPL definition for the X.25 NPSI server
- NPSI definitions
- VTAM SWNET definitions for NPSI

The primary diagnostic information source is the activity log produced by the X.25 NPSI server. Messages appear in the MVS system log, and can also be captured into a separate data set by including a SYSPRINT DD statement in the X.25 NPSI cataloged procedure. Normal logging records virtual circuit establishment and termination.

Additional information can be recorded about VC activity by setting the TRACE CONTROL option in the X.25 NPSI server configuration data set. This level is sufficient for almost all problem situations; interpretation of the data requires knowledge of X.25 NPSI packet formats. Tracing of the contents of IP datagrams sent to and received from NPSI is provided by the MVS CTRACE option. For details on using the CTRACE option, see Chapter 5, “TCP/IP services traces and IPCS support” on page 53.

VTAM buffer traces and NPSI X.25 line traces can also be useful in diagnosing difficult problem situations.

The IP packet trace facility can be used to trace the flow of IP packets. It is useful when tracking the cause of packet loss or corruption. See Chapter 5, “TCP/IP services traces and IPCS support” on page 53, for details about how to use the IP packet trace facility.

X.25 trace examples

The message severity codes (last position of the message ID) are:

I	Information (including trace)
W	Warning
E	Recoverable error
S	Recoverable error
T	Irrecoverable error

The following example shows normal initialization:

```
EZB2111I VTAM ACB X25IPI1 opened successfully
EZB2210I MCH XU038 packet level ready
EZB2451I IP AS path accepted for job name TCPIPTES
```

Initialization has four main steps:

1. The configuration file is read and processed.
2. VTAM control blocks are initialized (EZB2111I).

3. NPSI physical links (MCHs) configured by LINK statements are initialized (EZB2210I).
4. TCPIP establishes an IUCV path to the X.25 NPSI server (EZB2451I).

Normal incoming call, TRACE OFF

The following example illustrates a normal incoming call with TRACE OFF:

```
EZB2301I VC F001XU038    incoming call from 00000039 user data CC
EZB2325I VC F001XU038    facilities: pkt1024.
EZB2320I VC F001XU038    NPSI logon LU VL038001
EZB2330I VC F001XU038    call complete
    ...some time later...
EZB2350I VC F001XU038    call cleared, cause=00 diagnostic=C5
EZB2351I VC F001XU038    connection terminated for 00000039: sent 1 received
1 dropped 0
EZB2352I VC 010 closed
```

Notes:

1. The VC identifier F001XU038 ties together the events associated with a single virtual circuit. Messages for one VC are usually intermixed with messages for other VCs.
2. The X.25 address originating the call (00000039) is reported in the EZB2301I message.
3. X.25 calls can optionally request facilities to be applied, such as window size, packet size, throughput class, and reverse charging. These are reported in the EZB2325I message.
4. EZB2330I “call complete” indicates the virtual circuit is ready for transferring TCP/IP data.
5. An X.25 call can be closed by the originator, the acceptor, or the X.25 network. The cause and diagnostic codes in the EZB2350I message indicate the reason. In the example, cause=00 indicates the originator has closed the connection. Lists of cause and diagnostic codes can be found in *X.25 NPSI Diagnosis, Customization, and Tuning*.
6. EZB2351I reports the number of IP datagrams transferred on the virtual circuit.
7. After the EZB2352I “closed” message is issued, the virtual circuit is ready for reuse by another incoming call or to originate a new call.

Normal incoming call, TRACE DATA

The following example illustrates a normal incoming call with TRACE DATA:

```
EZB2230I MCH XU038    packet received (length=17)
EZB2000I 0000 .0.h..... 0BF00188 00000038 00000039 03420A0A
EZB2000I 0010 .    CC
EZB2301I VC F001XU038    incoming call from 00000039 user data CC
EZB2325I VC F001XU038    facilities: pkt1024.
EZB2302I VC F001XU038    call accept packet sent (length=6)
EZB2000I 0000 .0.... 0FF00102 0400
EZB2320I VC F001XU038    NPSI logon LU VL038001
EZB2330I VC F001XU038    call complete

EZB2332I VC F001XU038    data received (length=276)
EZB2000I 0000 E.....<.....} 45000114 00100000 3C017F82 820FFD26
EZB2000I 0010 ..}...*k.:wxr-(. 820FFD11 0800AA6B 00BAF778 72ADA88E
EZB2000I 0020 0}.f9kq.,.PF;._n 307D0C66 B96BF118 AC085046 3B83DF6E
    ....data omitted for brevity...
EZB2000I 0110 = 3.    BD5F339D
EZB2331I VC F001XU038    data sent (length=277)
EZB2000I 0000 .E.....<.....} 00450001 14001000 003C017F 82820FFD
EZB2000I 0010 ..};&.;2k.:wxr-( 11820FFD 260000B2 6B00BAF7 7872ADA8
EZB2000I 0020 .0}.f9kq.,.PF;._ 8E307D0C 66B96BF1 18AC0850 463B83DF
```

```

....data omitted for brevity...
EZB2000I 0110 _=_3.          5FBD5F33 9D

EZB2336I VC F001XU038      inactivity timer expired
EZB2353I VC F001XU038      clear request packet sent (length=5)
EZB2000I 0000 .....      00011300 00
EZB2365I VC F001XU038      clear sent
EZB2333I VC F001XU038      packet received (length=1)
EZB2000I 0000 .           17
EZB2358I VC F001XU038      clear confirmed
EZB2351I VC F001XU038      connection terminated for 00000039: sent 1
                             received 1 dropped 0
EZB2352I VC 010 closed

```

TRACE DATA can be used to record the full contents of IP datagrams as they pass through the X.25 NPSI server. The IP header begins at byte 45 (X'2D') within the IP packet. A reduced trace given by TRACE CONTROL shows only the X.25 control packets (call request, call accept, clear request, and clear confirm). Refer to *X.25 NPSI Host Programming* for the detailed packet formats.

Normal outgoing call, TRACE CONTROL

The following example illustrates a normal outgoing call with TRACE CONTROL:

```

EZB2310I VC F810XU038      outgoing call to 00000039
EZB2311I VC F810XU038      call request packet sent (length=20)
EZB2000I 0000 .....h..... 0B081002 04008800 00003900 00003803
EZB2000I 0010 .....      420A0ACC
EZB2230I MCH XU038         packet received (length=5)
EZB2000I 0000 ...0.        0F0810F0 01
EZB2314I VC 0810XU038      call accepted by user data
EZB2320I VC 0810XU038      NPSI logon LU VL038001
EZB2330I VC 0810XU038      call complete

EZB2336I VC 0810XU038      inactivity timer expired
EZB2353I VC 0810XU038      clear request packet sent (length=5)
EZB2000I 0000 .....      00011300 00
EZB2365I VC 0810XU038      clear sent
EZB2333I VC 0810XU038      packet received (length=1)
EZB2000I 0000 .           17
EZB2358I VC 0810XU038      clear confirmed
EZB2351I VC 0810XU038      connection terminated for 00000039: sent 5
                             received 5 dropped 0
EZB2352I VC 010 closed

```

The steps involved in outgoing and incoming calls are similar. One important difference is that the virtual circuit identifier changes when the call is accepted (compare the EZB2311I and EZB2314I messages). This is related to the details of the NPSI programming interface.

X.25 experts should note that some X.25 packets do not appear in the trace because they are generated by NPSI without the direct involvement of the host application. Clear confirm is one example. Also, the sequence of events during closing can vary slightly in normal operation, and in some instances, benign VTAM request failures can be reported with message EZB2411E.

Results of LIST command

The following example illustrates the results of the LIST command:

```

EZB2020R MCH XU038      state 1050
EZB2021R VC 010 LU VL038001 DTE 00000039      state 4050
EZB2021R VC 00F LU      DTE      state 1010
...
EZB2021R VC 001 LU      DTE      state 1010
EZB2022R IP AS TCPIPTES state 80

```

The LIST command is useful to get a snapshot of virtual circuit status. This example shows a normal status with one active VC (state 4050). VC state 1010 indicates ready but not in use. With the NPSI fast connect feature, the normal idle state is 1050. Other intermediate states can appear while an X.25 call or clear is in progress. The codes are listed in *z/OS Communications Server: IP Messages Volume 1 (EZA)*.

The status of the IUCV path to TCPIP is shown in the last line; 80 is normal; 00 indicates that the TCPIP X25 NPSI device has not been started.

Termination by TCPIP STOP device

The following example illustrates termination using the TCPIP STOP device:

```

EZB2091I HALT notice accepted, type 0
EZB2250I MCH XU038      terminating
EZB2352I VC 010 closed
EZB2352I VC 00F closed
...
EZB2352I VC 001 closed
EZB2480I IP AS TCPIPTES disconnected: sent 7 received 7 dropped 0
EZB2090I Terminating
EZB2099I Ended

```

EZB2480I reports the number of IP datagrams transferred on the IUCV path for TCPIP.

Logon problems

Several steps must take place successfully to establish an X.25 virtual circuit for TCP/IP activity:

1. An X.25 call request is received by the X.25 NPSI server from the X.25 network (incoming call) or is sent by the X.25 NPSI server to establish a connection to a new destination (outgoing call).
2. An X.25 call accept confirms the X.25 call request. Call accept is sent by TCPIPX25 for an incoming call, or received from the X.25 network for an outgoing call.
3. NPSI initiates an SNA session with the X.25 NPSI server application by means of a VTAM LOGON.

Each of these steps is reported in the activity log, shown in the “X.25 trace examples” on page 646. Problems fall into two main areas: failure of the X.25 call itself, indicated by either a refusal or an immediate clear, or failure of the NPSI LOGON. Call failures are reported with X.25 cause and diagnostic codes. Standardized cause codes include:

Code Meaning

- | | |
|-----------|---|
| 00 | DTE clearing. The remote system cleared the call. |
| 01 | Number busy. The called number cannot accept another call. |
| 03 | Invalid facility request. A facility requested by the caller is not subscribed or conflicts with a subscribed option. |

- 05** Network congestion. Congestion conditions or some other problem within the network temporarily prevent the requested virtual circuit from being established.
- 09** Out of order. The called number is out of order.
- 0B** Access barred. The caller is not permitted to obtain a connection to the called number.
- 0D** Not obtainable. The called number is not assigned or is no longer assigned.
- 11** Remote procedure error. An X.25 protocol error at the remote equipment.
- 13** Local procedure error. An X.25 protocol error.

Refer to *X.25 NPSI Diagnosis, Customization, and Tuning* for a list of diagnostic codes. X.25 networks can also have special diagnostic codes in the range 80–FF.

VC LOGON can fail for a variety of reasons. Among the most common reasons are:

- Incorrect VTAM switched circuit definitions. IDNUM entries are error prone; SSCPFM=USSNTO and DISCNT=(YES,F) are necessary.
- A default VTAM USS table ISTINCDT that has been modified to include text in the message 10 entry.
- Coding LOGAPPL on the NPSI X25.VC definitions. LOGAPPL should only be used on the X25.MCH and on the X25.VC with the Fast Connect feature.
- Insufficient number of type 1 LUs configured on the NCP LUDRPOOL statement.

A VTAM buffer trace with ID=VTAM will help diagnose the first problem. Collect the following configuration documentation before contacting the IBM Software Support Center. X.25 NPSI server configuration data set, VTAM APPL definition for the NPSI X.25 server, NPSI definitions, and VTAM SWNET definitions for NPSI.

Session hangs

In diagnosing session hang or timeout problems, remember that TCPIPX25 does not track individual TCP sessions; it only transfers IP datagrams. One X.25 virtual circuit can carry datagrams from several TCP sessions. A VC can also be closed and reestablished several times during a TCP session with long periods of inactivity. Failure of an X.25 connection is not directly reflected in TCP sessions it might be carrying, only indirectly by TCP timeouts.

Opening a TCP session, such as a Telnet connection, can fail for reasons not specific to X.25, for example, a TCPIP routing problem caused by an incorrect GATEWAY definition, or an IP routing problem in the remote device. Symptoms suggesting these problems include:

- No X.25 call is made when a TCP connection is requested.
- No traffic is received from the remote equipment, indicated by a received count of zero in the EZB2351I connection terminated message.

An established TCP connection can hang because the X.25 network or remote device is down. This will be indicated by a clear cause and diagnostic, as described in “Logon problems” on page 649.

Helpful hints

PING fails but Telnet and FTP connect. Setting up a new X.25 connection might take longer than the default PING timeout on a busy system. Use the PING

TIMEOUT or COUNT parameters to extend the waiting time. Use the NPSI GATE Fast Connect feature to reduce connection setup time.

PING succeeds but Telnet or FTP data transfer times out. Full-screen Telnet and FTP data transfers create large IP datagrams, while PING uses smaller ones. If the small datagrams go through but large ones do not, there might be a problem with MAXDATA on the VTAM switched circuit definitions; see “Configuration requirements” on page 645 for details. Attempting to pass a datagram larger than MAXDATA on a virtual circuit will hang the VC for all subsequent traffic.

A load-dependent hang can be due to an insufficient number of virtual circuits.

The TRAFFIC command can be used to observe virtual circuit data transfer activity.

Documentation requirements

If IBM Support Center help is needed, collect the following configuration documentation before contacting IBM:

- X.25 NSPI server console log showing X.25 connections related to the problem
- X.25 NSPI server configuration data set
- PROFILE.TCPIP data set
- NPSI definitions
- VTAM SWNET definitions for NPSI

Chapter 28. Diagnosing IMS problems

The IMS TCP/IP Services socket interface allows TCP/IP clients to access IMS using a TCP/IP network. This access is fully described in the *z/OS Communications Server: IP IMS Sockets Guide*. A sockets program-to-program connection is established between a client (TCP/IP socket) program and a server (IMS application) program. TCP/IP and the Listener are agents in the connection establishment. The components of the IMS TCP/IP socket interface system are shown in Figure 73.

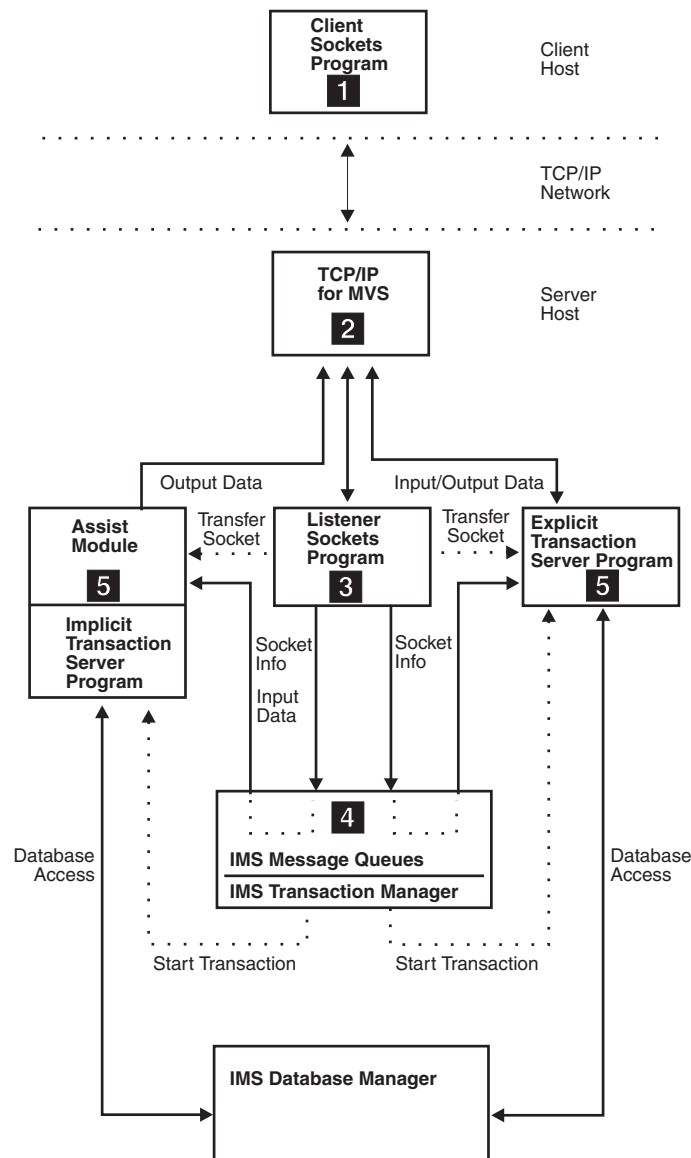


Figure 73. Components of the IMS TCP/IP services socket interface system

The following list is a brief description of the component interaction and data flow that occurs when a client program requests an IMS transaction.

- 1** The client program starts and sends the transaction request message (TRM) to the Listener port.
- 2** The Listener reads the TRM and accepts the socket connection between the client program and the Listener from TCP/IP.
- 3** The Listener validates the TRM, prepares to give the socket connection to the IMS transaction, builds the transaction initiation message (TIM) containing the socket connection information, and sends the TIM to the IMS transaction manager message queue. For implicit IMS transactions, the Listener also reads the input data from the client program and sends it to the message queue.
- 4** The IMS transaction manager schedules the requested transaction.
- 5** IMS Transaction. This can be one of the following:

Implicit

The IMS assist module receives the TIM on behalf of the implicit IMS transaction and takes the socket connection from the Listener. The input data is read and the IMS transaction performs the required database access. The IMS assist module, on behalf of the implicit IMS transaction, writes the output data to the client program, through the socket connection, followed by the commit status message (CSM). The socket connection then closes.

Explicit

The explicit IMS transaction receives the TIM and takes the socket connection from the Listener. Input and output data is read and written as defined by the protocol, and the required database access is performed. The explicit IMS transaction writes the CSM to the client program and closes the socket connection.

The IMS transaction and the client program terminate.

Setting up the IMS TCP/IP services socket interface system

Completing the following steps establishes the system described in Figure 73 on page 653. Each step should be completed in order, from the first step to the last.

This list of steps can be used to diagnose problems in starting components by identifying the prerequisites. The steps immediately preceding a step in which you are told to start a component are required to give definitions and configuration information that must be completed correctly before that component can be started. The reference keys in the steps refer to the components as shown in Figure 73 on page 653. All components except the client sockets program belong to the server host.

1. Configure TCP/IP to reserve the Listener port number.
A TCP/IP port should be reserved for the Listener to connect to when it starts. The following is a sample profile statement to reserve the Listener port.

```
PORT 4096 TCP EZAIMSLN
```

Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for details about the PORT statement.

2. Configure the TCP/IP network from the server host to the client host.
For the client program to issue IMS transaction requests across a socket connection, there must be a TCP/IP network defined between the client and

server hosts. Any physical network supported by IBM MVS TCP/IP can be used to establish this socket connection.

Refer to the appropriate chapters in the *z/OS Communications Server: IP Configuration Reference* for details about how to configure the required network to the server host TCP/IP.

3. **2** Start the TCP/IP address space on the server host.
4. Establish and verify the network connection from the client host to the server host.

Depending on the network connection, start or activate the required device drivers and network nodes required to establish a TCP/IP network connection.

To verify the TCP/IP network connection, use the PING command on the client host, using the server host destination IP address or network name.

5. Define the Listener to the IMS transaction manager.

The IMS transaction manager must be defined to expect message queue input from the Listener. For information about how to define the Listener to IMS, refer to the Listener IMS definitions in the *z/OS Communications Server: IP IMS Sockets Guide*.

6. **5** If the IMS transaction that will be requested by the client program is not already written, write it.

Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for specific details about writing IMS transactions that can be requested by a TCP/IP client program.

7. Define the IMS transaction that will be requested by the client program to the IMS transaction manager.

The IMS transaction must be defined to IMS before the Listener can request it to be scheduled on behalf of the client program. Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for important restrictions when defining IMS transactions.

8. **4** Start the IMS transaction manager and the IMS database manager.
9. Complete the Listener configuration data set.

The Listener configuration data set is read when the Listener is started. The procedure used to start the Listener (usually EZAIMSLN) uses the ddname LSTNCFG to specify the Listener configuration data set. Following is an example statement that specifies TCPIP.LISTENER.DATA as the configuration data set.

```
LSTNCFG DD DSN=TCPIP.LISTENER.DATA,DISP=SHR
```

This data set must contain a minimum set of required statements to specify the environment the Listener is started in and the list of IMS transactions available to client programs.

Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for details about the format and contents of this data set.

10. **3** Start the Listener address space.

The Listener is started as an MVS address space as described in the *z/OS Communications Server: IP IMS Sockets Guide*. The JCL procedure required for starting the address space is also listed in the *z/OS Communications Server: IP IMS Sockets Guide*.

11. Write the client program, if not already written.

Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for programming details about client programs that can request IMS transactions over a TCP/IP network.

12. **1** Start the client program.

Common configuration mistakes

The following is a list of common configuration mistakes:

- The IMS transaction has not been defined in the Listener configuration data set.
- The Implicit or Explicit parameter in the Listener configuration data set does not match the protocol used by the IMS transaction.
- The program specification block (PSB) for the Listener does not include the ALTPCB label.
- The IMS transaction invoked by the Listener does not specify the MODE=SNGL parameter on the IMS TRANSACT macro in the IMS database manager definition. Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for information about restrictions on application programs.
- The IMS transaction invoked by the Listener was not defined to the IMS transaction manager as a multisegment transaction.
- The IMS transaction invoked by the Listener is an IMS conversational transaction or executes in a remote Multiple Systems Coupling (MSC) environment.

Quick checklist for common problems

The following list summarizes some initial checks that can be made quickly and are helpful in identifying the problem area.

1. Is the TCP/IP network active?
To verify that the network to the server host is active, use the PING command on the client host, using the same IP address or host name as specified in the client program.
2. Is the Listener started and active on the server host?
Check that the Listener address space is active and running. The MVS SDSF facility can be used to view the active address space list. Also see “Using NETSTAT” on page 667 for details about how to determine if the Listener TCP/IP port is active.
3. Did the Listener program list any configuration errors to the SYSPRINT data set?
Check the JCL DD statement in the Listener start procedure to identify the destination of the SYSPRINT output. See “Where to find error message documentation” on page 669 to determine the reason for any errors. The Listener address space might need to be stopped to flush any error messages to the destination.
4. Have you completed all of the required definitions. See “Setting up the IMS TCP/IP services socket interface system” on page 654 for the list of required definitions and configurations.
5. Is the client program connecting to the same TCP/IP port as the Listener? See “Using NETSTAT” on page 667 for details about how to use the NETSTAT command to identify which port the Listener is connected to and which port the client program is establishing a socket connection on.

Component problems

These problems are related to starting or stopping one of the components in the IMS TCP/IP Services socket interface system.

- *The Listener terminates on startup.*

The following is a list of possible causes and resolutions.

Cause Incorrect configuration data set.

Resolution

Check for configuration error messages written to the SYSPRINT data set and correct the problems (if any).

Cause The prerequisites for starting the Listener have not been completed.

Resolution

Complete the required steps listed in “Setting up the IMS TCP/IP services socket interface system” on page 654.

Cause Incorrect method of starting.

Resolution

Ensure the Listener is being started as an MVS address space as described in the *z/OS Communications Server: IP IMS Sockets Guide*. The JCL procedure required for starting the address space is also listed in *z/OS Communications Server: IP IMS Sockets Guide*.

- *The Listener will not terminate.*

Cause The Listener will wait for all of the currently open socket connections to close before it responds to the user termination request. If any of the socket connections have hung, the Listener needs to be forcibly terminated.

Resolution

Force the Listener to terminate using the command specified in the section about stopping the IMS Listener in the *z/OS Communications Server: IP IMS Sockets Guide*.

See “Connection problems” on page 658 for a description of how socket connections can hang.

- *As the Listener is starting, messages are written to the system console asking if IMS should be started.*

Cause The IMS system should be started before the Listener. If the Listener is started first, the operator is prompted to start the IMS system.

Resolution

Reply to the console messages to start IMS.

- *An implicit IMS transaction written in C is experiencing unexpected problems at startup.*

Cause If IMS transaction programs written in C are not built correctly, the IMS interface will fail on startup.

Resolution

Build the C program correctly as specified in the section about writing an IMS TCP/IP Services server program in *z/OS Communications Server: IP IMS Sockets Guide*.

- *The Listener is abending while accepting the TRM.*

Cause If a user-defined security exit has been linked into the Listener, it might be causing the problem. The security exit is called when validating the

TRM. If the security exit has not been written to accept the required linkage and parameters, the Listener will abend because the exit runs in the same address space.

Resolution

Check that the security exit has been written to accept the linkage and parameters as specified in the section on the IMS security exit in the *z/OS Communications Server: IP IMS Sockets Guide*.

Connection problems

These problems are related to the TCP/IP socket connection. They include problems with establishing the connection, transferring data over the connection, and unexpected loss of the connection.

- *The client program is experiencing intermittent reject connect responses from TCP/IP.*

Cause The TCP/IP sockets facility has a connection request backlog queue. While this queue is full, further connection attempts will be rejected by TCP/IP. Under load, this queue can temporarily fill, causing some client program requests to be rejected.

Resolution

To reduce the frequency of this problem, increase the size of the backlog queue. The size of the queue is controlled by a parameter in the Listener configuration data set.

- *The TCP/IP socket connection to the client program is being broken immediately after the implicit IMS transaction is scheduled.*

Cause The Listener configuration data set might incorrectly define the implicit IMS transaction as explicit. In this case, the Listener will not pass the input data to the IMS transaction through the message queue as expected. The transaction will start, and upon detecting no data, immediately close the TCP/IP socket connection and terminate.

Resolution

Verify that the TRANSACTION statements in the Listener configuration data set specify the TYPE parameter correctly.

- *Connection lockup for an implicit IMS transaction.*

A connection lockup occurs when both the implicit IMS transaction and the client program are waiting for data from the other end of the socket connection.

Cause The Listener might be waiting for the end-of-message (EOM) segment from the client program. The client program must send a valid EOM segment before the Listener will instruct the IMS transaction manager to schedule the IMS transaction. If the client program does not send a recognized EOM segment, the Listener waits indefinitely for it, while the client program waits for a response.

Resolution

Use the IP packet trace facility to determine whether the client program is sending a valid EOM segment. See “Using IP packet trace” on page 666 for details about the IP packet trace facility.

Refer to the information about implicit-mode application data in the *z/OS Communications Server: IP IMS Sockets Guide* for the format of the EOM segment.

- *Connection lockup for an explicit IMS transaction.*

A connection lockup occurs when both the explicit IMS transaction and the client program are waiting for data from the other end of the socket connection.

Cause Because the explicit IMS transaction protocol is user defined, programming errors can easily lead to connection deadlocks. That is, the server is waiting for more data while the client is waiting for a response, and both will wait indefinitely.

Resolution

Use the IP packet trace facility to identify which part of the protocol is failing. See “Using IP packet trace” on page 666 for details about the IP packet trace facility.

Cause The Listener configuration data set might incorrectly define the explicit IMS transaction as implicit. In this case the Listener will wait for valid implicit data from the client program, or if valid data is received, the explicit IMS transaction will wait for data from the client program because the Listener has already read the data and written it to the message queue.

Resolution

Verify that the TRANSACTION statements in the Listener configuration data set specify the TYPE parameter correctly.

Timeouts, especially in the client program, are recommended when issuing socket READs to avoid deadlocks and allow easy diagnosis. Refer to the information about SELECT calls in the *z/OS Communications Server: IP IMS Sockets Guide* for more information about specifying timeouts for READs.

- *Connection lockup for either an explicit or implicit IMS transaction.*

A connection lockup occurs when both the IMS transaction and the client program are waiting for data from the other end of the socket connection.

Cause If the TRM sent by the client program is incomplete, the Listener will wait indefinitely for the rest of the message.

Resolution

Check the length and format of the TRM by using the IP packet trace facility as described in “Using IP packet trace” on page 666.

Cause If the IMS transaction does not successfully issue the takesocket to gain the connection from the Listener, the Listener will wait for this event indefinitely. The takesocket might not be issued successfully due to one of the following reasons:

- The IMS transaction is defined to run in a message processing region that is not started. In this case, the IMS transaction will never be scheduled and, therefore, never issue the takesocket.
- One of the several TCP/IP socket calls, up to and including the takesocket, might fail and terminate the IMS transaction.
- An IMS error can stop the transaction from being successfully scheduled, or, especially in the explicit case, can cause the IMS transaction to terminate before the takesocket is issued.

Resolution

Check that the IMS transaction is being successfully scheduled by the IMS transaction manager and ensure that any IMS and socket calls issued by the IMS transaction are checked for unsuccessful return codes.

- *The takesocket call issued by the IMS transaction fails.*

Note: For implicit transactions, the IMS assist module routines issue a takesocket for the first get unique (GU) issued by the transaction. If the takesocket fails, the GU returns ZZ.

Cause IMS can, for recovery reasons, abend a transaction and start it again. If the transaction is abended after it has gained the socket connection (through a takesocket call), the TCP/IP socket connection is lost. Although IMS restores the message queue when it restarts the transaction, the takesocket issued by the transaction will fail as the socket connection has already been taken from the Listener.

Resolution

Restart the client program. To reduce the frequency of this problem, determine why IMS is restarting the IMS transaction by using the IMS trace facility. See “IMS traces” on page 667.

Cause An IMS transaction not defined as multisegment to the IMS transaction manager will be scheduled as soon as the TIM is added to the message queue. This gives the IMS transaction an opportunity to issue the takesocket before the givesocket is issued by the Listener. The takesocket will fail with an error return code.

Resolution

Make certain the IMS transaction is defined as multisegment.

- *The client program is always receiving reject connect responses from TCP/IP.*

Cause The maximum number of active sockets might have been reached, with all the currently active socket connections unable to complete. An increasing number of socket connections will eventually reduce the available socket connections to zero when the number of socket connections equals the MaxActiveSockets configured for the Listener. When this happens, TRMs are not processed by the Listener, and they are left on the TCP/IP backlog queue. When the backlog queue fills, TCP/IP will reject a client program connection attempt.

Resolution

Identify the client programs causing the problem using the NETSTAT command as specified in “Using NETSTAT” on page 667; then continue diagnosis to determine why these connections are locking up.

The Listener must be restarted to clear the active socket list. Because there are active socket connections, the Listener must be forced to terminate using the command specified in the *z/OS Communications Server: IP IMS Sockets Guide*.

- *Connection lockup or loss when passing a socket connection from one explicit IMS transaction to another.*

A connection lockup is when the socket connection reaches a state where it will never complete.

Cause To pass a socket connection from the first IMS transaction to the second, the first IMS transaction must wait after it issues the givesocket until the second IMS transaction issues a takesocket; otherwise, the connection will be lost.

A connection lockup can occur when the first IMS transaction waits for the takesocket to be issued, but both IMS transactions are defined to run in the same message processing region. In this case, they cannot both be scheduled to run at the same time, and the first IMS transaction will wait indefinitely for the takesocket from the second IMS transaction, which will never be scheduled.

Resolution

When passing a socket connection between IMS transactions, make sure the first transaction waits for the second to issue the takesocket and that both IMS transactions can be scheduled to run at the same time.

Error message and return code problems

These problems are related to error responses.

- *The client program is receiving a request status message (RSM).*

Cause The Listener will send this message to the client program when it detects an error condition.

Resolution

Use the return and reason codes from the message to look up the explanation. See “Where to find return code documentation” on page 668.

- *The implicit IMS transaction is receiving return codes in the I/O program communication block (PCB) that are not defined in the section on status codes in the IMS/ESA Diagnosis Guide and Reference.*

Cause The IMS assist module will perform several socket-related functions on behalf of the implicit IMS transaction in response to IMS transaction manager requests. When errors are detected that are not related to the IMS transaction manager request, the IMS assist module sets special return codes in the PCB.

Resolution

Look up the meaning of the special return codes. See “Where to find return code documentation” on page 668.

- *The Listener error messages are written to the MVS system console instead of the SYSPRINT data set.*

Cause If the Listener experiences data set I/O errors, it will redirect the error messages to the MVS system console.

Resolution

Check the MVS system console log for I/O errors on the data set to identify the problem. The SYSPRINT DD statement in the JCL procedure to start the Listener specifies the destination data set for the error messages.

Socket data protocol problems

These problems are related to data transfer over the socket connection. They include incorrect data sent, not enough or too much data sent, and data corruption.

- *The Listener is not responding to the client program.*

Cause If the TRM sent by the client program is incomplete, the Listener will wait indefinitely for the rest of the message.

Resolution

Check the length and format of the TRM by using the IP packet trace facility as described in “Using IP packet trace” on page 666.

Cause If the port specified by the client program is not the port that is attached to the Listener, and the socket connection is established, the other end of the connection will not communicate with the client program as required.

Resolution

Check that the Listener is attached to the port used by the client program to establish the socket connection. Use the command specified in “Using NETSTAT” on page 667.

- *All the input data sent from the client program is not being passed to the implicit IMS transaction from the Listener.*

Cause Any input data written after the first EOM segment will be ignored by the Listener.

Resolution

Check for EOM segments being sent by the client program by using the IP packet trace facility described in “Using IP packet trace” on page 666.

Refer to the information about the implicit-mode application data in the *z/OS Communications Server: IP IMS Sockets Guide* for the format of the EOM segment.

- *Explicit IMS transaction is receiving garbled data from or sending garbled data to the client program.*

Cause The data might need translation when the client program does not exist on an EBCDIC host. For explicit data transfer, the client program, or the IMS transaction, or both, must provide ASCII to EBCDIC translation and byte-order translation of fixed-point binary integers, if required. The Listener automatically translates the TRM when creating the TIM.

Resolution

Code the client program or the IMS transaction or both to provide the necessary translation when the client program is not on an EBCDIC host.

- *Implicit IMS transaction is receiving garbled data from or sending garbled data to the client program.*

Cause The automatic data translation when the client program does not exist on an EBCDIC host can be causing the problem. For implicit data transfer, the Listener automatically translates input data from ASCII to EBCDIC, based on the TRM contents. The IMS assist module also automatically translates output data from EBCDIC to ASCII when sending to an ASCII client program, as determined by the TRM. If the TRM sent by the client program is not either ASCII or EBCDIC as required, then the automatic translations fail. The client program is also responsible for any required byte-order translation of fixed-point binary integers.

Notes:

1. If the data translated between ASCII and EBCDIC contains any nonprintable data, such as integers, flags, or reserved fields, the data will be corrupted. In this case, the client program must provide EBCDIC data (including the TRM) for the IMS transaction and expect EBCDIC data from the IMS transaction.
2. If the data is translated between ASCII and EBCDIC and contains characters that are not common to both the ASCII and EBCDIC tables, the nontranslatable characters will be translated to spaces.

Resolution

Code the client program to provide the necessary translation when the client program is not on an EBCDIC host and the automatic data translation cannot be used.

- *The security exit will not validate user data from the client program.*

Cause The security exit might not be successfully linked into the Listener. The exit must be compiled and assembled and then linked into the Listener for it to be called.

Resolution

Check that the security exit has been coded and built correctly as specified in the *z/OS Communications Server: IP IMS Sockets Guide*.

- *Data is corrupted after an implicit IMS transaction issues a GU.*

Cause The I/O area declared might be too small. When using the IMS assist module, the I/O area provided for the GU call must be large enough to hold the TIM, even though the data eventually returned in the I/O area can be smaller.

Resolution

Make certain the implicit IMS transaction has enough storage declared to hold the TIM. The size of this message is specified in the *z/OS Communications Server: IP IMS Sockets Guide*.

- *The PL/I IMS transaction is receiving or sending message segments that are not valid.*

Cause The message segments might be declared incorrectly. The PL/I API interface to the IMS transaction manager defines the message segments with a four-byte length field, but the length value must include only two of those bytes plus the rest of the segment.

Resolution

Use the following rules to avoid problems:

- The IMS assist module PL/I API routines mimic the interface used by the PL/I API routines. Code PL/I implicit transaction message segments in exactly the same manner as for this interface.
- Code the client program in exactly the same manner as for all the IMS transaction API interfaces. The IMS assist module routines will automatically convert the message segments from the PL/I API to the standard format.
- Explicit transactions do not use the IMS assist module. The message segment format, if required, must match on both the client program and the IMS transaction sides. It is recommended that the standard message segment format be used.

Refer to the information about programming considerations for the implicit-mode server and the explicit-mode server in the *z/OS Communications Server: IP IMS Sockets Guide* for more details about the PL/I API issues.

IMS transaction build problems

This problem relates to building a component in the IMS TCP/IP Services socket interface system.

Unresolved external reference errors are causing the linker to fail when linking an IMS transaction.

Cause The implicit IMS transaction link JCL is not including the IMS assist module and the MVS TCP/IP Services sockets library to resolve external references.

Resolution

Compare the link JCL to the sample provided in the section about JCL for linking an implicit-mode server in the *z/OS Communications Server: IP IMS Sockets Guide*.

Cause The explicit IMS transaction link JCL is not including the MVS TCP/IP Services sockets library to resolve external references.

Resolution

Compare the link JCL to the sample provided in the section about JCL for linking an explicit-mode server in the *z/OS Communications Server: IP IMS Sockets Guide*.

IMS database problems

These problems relate to unexpected IMS database actions or failures. They include changes not made or requests for changes that fail.

- *The IMS transaction is terminating without performing the required function and without issuing any error messages.*

Cause The IMS transaction might not be checking for interface errors.

Resolution

It is the responsibility of the IMS transaction programmer to identify and issue error messages if the IMS database manager, IMS transaction manager, or TCP/IP socket interfaces fail.

- *The client program is not receiving any data from the implicit IMS transaction, but is receiving a successful CSM.*

Cause The IMS transaction might be issuing an IMS database rollback (ROLB) call. If the IMS transaction issues a ROLB call, all output accumulated by the IMS assist module is discarded as part of the ROLB function. Depending on how the IMS transaction is coded, it might complete without further output (ISRT calls).

Resolution

Use caution in issuing ROLB calls in implicit IMS transactions using the IMS assist module. Make certain you understand the details about implicit-mode support for ROLB processing in the *z/OS Communications Server: IP IMS Sockets Guide*.

- *Local IMS transaction manager ISRT/GU/GN calls are failing when issued in IMS transactions.*

Cause Local calls assume a terminal has requested the IMS transaction. The input and output of data, however, is actually sent across the socket connection for IMS transactions started by the Listener. The following is a list of specific causes of the problem:

- The ISRT call has no terminal associated with the IMS transaction for the output.
- There is no data on the message queue for explicit IMS transactions to get with the GU or GN calls.
- An implicit IMS transaction will receive an unexpected TIM in response to a GU call.

Resolution

Do not issue local IMS transaction manager calls from transactions started by the Listener. An implicit IMS transaction must use the IMS

assist module calls, which will access either a terminal or socket connection, as required. An explicit IMS transaction must interface directly to the socket connection.

- *The ISRT call fails for an implicit IMS transaction if a large amount of data is output.*

Cause The IMS assist module restricts the total output for a single IMS transaction execution to 32KB.

Resolution

Limit the output for an implicit IMS transaction using the IMS assist module to a total of 32KB.

- *The IMS database manager commits the changes made by an IMS transaction, but the client program receives an error.*

Cause The implicit IMS transaction does not issue a second GU. The IMS database commits the changes either when the IMS transaction ends or when another GU is issued. For implicit IMS transactions, the IMS assist module routines sends the output data and CSM to the client program and closes the socket connection when the second GU is issued. If the implicit IMS transaction does not issue another GU, the changes are committed when the transaction ends, but the client program assumes failure when the CSM is not received.

Resolution

Implicit IMS transactions that are started by the Listener must issue GU calls to get the next transaction request until the GU call returns with no requests to process.

Cause The socket connection might have been broken after the changes were committed but before the CSM was sent. In this case, the client program will assume failure, but the changes have been committed.

Resolution

Where possible, the client program should be coded to automatically restart the IMS transaction and handle the condition where the IMS transaction is duplicated. For explicit IMS transactions, a more rigorous protocol can be implemented.

Note: This should be considered as an uncommon case.

- *The client program does not receive a valid CSM from an implicit IMS transaction.*

Cause The client program might not have completed the response protocol correctly. The client program must read the response data until it reads an EOM segment. The CSM immediately follows the EOM.

Resolution

Use the IP packet trace facility to determine whether the IMS transaction is sending a valid EOM segment followed by a valid CSM segment. See "Using IP packet trace" on page 666 for details about the IP packet trace facility. If the correct message segments are being sent, correct the client program to receive the response data.

Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for the format of the EOM and CSM segments.

Documentation references for problem diagnosis

This section contains the information and documentation references required to gather and decode diagnostic information about the IMS TCP/IP Services socket interface system.

The two main tools used for problem diagnosis are the IP packet trace facility and the NETSTAT utility. The use of these tools is explained in following sections and example statements and commands are provided. An explanation of how to interpret the output from each of these tools is also provided.

For TCP/IP or IMS-specific tracing, reference is made to the appropriate diagnosis documentation.

Two cross-reference sections, which list all the types of return codes and error messages that can be issued from the IMS TCP/IP Services socket interface system, are provided at the end of this section. For each type of return code and error message, a reference is made to existing documentation that provides a complete description.

Traces

The following traces can be used to gain information about data flows and actions of the IMS TCP/IP Services socket interface system. The IP packet trace facility is the most helpful trace facility when writing and debugging your own client programs and IMS transactions. The TCP/IP internal traces are mainly used to diagnose problems with the TCP/IP network and socket-specific problems. The IMS traces are mainly used to diagnose IMS-specific problems, such as IMS transaction scheduling and database commit and rollback errors.

Using IP packet trace

Use IP packet trace to identify the flow of data between the client program and the Listener and IMS transaction servers. TCP packets can be traced on the socket connections established through the Listener-reserved port. If the IP address of the client program is specified, only packets originating from or destined to the client program are traced. Specifying this parameter is recommended to avoid tracing a large number of unrelated TCP packets.

Note: When using X.25 devices to provide the network to the client program, the IP packet trace facility must be activated from the individual device address spaces. The previous example only activates tracing in the TCPIP address space.

See Chapter 5, “TCP/IP services traces and IPCS support” on page 53 for details about how to use the IP packet trace facility.

The packets that contain data display the data in hexadecimal digits and, in this case, their EBCDIC characters. The numeric fields in the message segments can be verified from the hexadecimal representation, while any alphabetic data can be verified from the EBCDIC display.

TCP/IP internal traces

The TCPIP internal traces are sent to CTRACE. These traces provide information about the internals of the TCPIP address space. This information can be used to diagnose problems in establishing the network between the client program and the

server host or in establishing the socket connections. See Chapter 5, “TCP/IP services traces and IPCS support” on page 53, for details about how to use the TCPIP internal tracing facility.

IMS traces

The IMS traces provide information about the internals of the IMS database system. This information can be used to diagnose IMS transaction scheduling problems, IMS transaction manager message queue problems, and database change problems that cause rollbacks or commit errors. For an overview of monitoring the IMS system, refer to *IMS/ESA Administration Guide: System*. For details about tracing and reading the trace reports refer to the *IMS/ESA Utilities Reference: System*.

Using NETSTAT

This section details how to use NETSTAT to query TCP/IP port usage and the state of socket connections. This command can be used to verify that the Listener is active and has opened the correct port and to diagnose problems with the socket connection between the client program and the Listener or IMS transaction.

Note: The client program must have the socket connection open for NETSTAT to query the connection status.

The NETSTAT SOCKETS command displays which ports are open to which address spaces and displays active socket connections and their status. Following is sample output from this command (the output shown is valid for V2R10 and V1R2):

```

READY
netstat sockets

MVS TCP/IP NETSTAT CS V2R10          TCPIP Name: TCPCS          12:34:56
Sockets interface status:
Type  Bound to          Connected to          State  Conn
=====
Name: INETD1    Subtask: 006DB5B8
Dgram 0.0.0.0..37      *..*                UDP    00000058
Dgram 0.0.0.0..13      *..*                UDP    00000057
Dgram 0.0.0.0..19      *..*                UDP    00000056
Dgram 0.0.0.0..9       *..*                UDP    00000055
Dgram 0.0.0.0..7       *..*                UDP    00000054
Stream 0.0.0.0..623     0.0.0.0..0          Listen 0000004B
Stream 0.0.0.0..514     0.0.0.0..0          Listen 0000004D
Stream 0.0.0.0..513     0.0.0.0..0          Listen 0000004C
Stream 0.0.0.0..512     0.0.0.0..0          Listen 0000004E
Stream 0.0.0.0..37      0.0.0.0..0          Listen 00000053
Stream 0.0.0.0..7       0.0.0.0..0          Listen 0000004F
Stream 0.0.0.0..13      0.0.0.0..0          Listen 00000052
Stream 0.0.0.0..19      0.0.0.0..0          Listen 00000051
Stream 0.0.0.0..9       0.0.0.0..0          Listen 00000050
Name: OSNMPD    Subtask: 006DBA70
Dgram 0.0.0.0..161      *..*                UDP    00000013
Stream 0.0.0.0..1027    0.0.0.0..0          Listen 00000014
Name: TCPCS    Subtask: 00000000
Stream 127.0.0.1..23     127.0.0.1..1033     Estblsh 00000045
Stream 9.67.113.27..23    9.37.81.207..1096   ClosWait 00000039
Name: TCPCS    Subtask: 006C57B0
Stream 0.0.0.0..23       0.0.0.0..0          Listen 00000012
Name: TCPCS    Subtask: 006D56F0
Stream 127.0.0.1..1026    127.0.0.1..1025     Estblsh 0000000F
Name: TCPCS    Subtask: 006D5CF0
Stream 0.0.0.0..1025     0.0.0.0..0          Listen 0000000C
Stream 127.0.0.1..1025    127.0.0.1..1026     Estblsh 00000010
Name: USER18    Subtask: 006A3400
Stream 127.0.0.1..1033    127.0.0.1..23       Estblsh 00000044

READY

```

Refer to *z/OS Communications Server: IP User's Guide and Commands* for more details about the usage, parameters, and output of NETSTAT.

Where to find return code documentation

The following list refers to the appropriate return code documentation for all return codes expected in the IMS TCP/IP Services socket interface system.

- To the client from the Listener (request status message).

Refer to the information about the request status message (RSM) segment in the *z/OS Communications Server: IP IMS Sockets Guide* for the format of the RSM and a description of the return codes.

Note: The RSM with the “IMS transaction unavailable to be started” return code, is returned when the IMS transaction has previously abended or failed and the IMS transaction manager has marked it as not able to be scheduled.

- To the client from an IMS transaction (CSM).

The CSM is received by the client program when the transaction is successful. This message implies a successful return code. If this message is not received, the client program must assume the IMS transaction has not completed successfully.

- To the implicit IMS transaction from the IMS assist module (I/O program communication block).

Refer to the information about the I/O PCB implicit-mode server in the *z/OS Communications Server: IP IMS Sockets Guide* for the format of the I/O PCB and return code explanations.

- To an implicit/explicit IMS transaction from TCP/IP.

Refer to the information about error messages and return codes for IMS sockets calls in the *z/OS Communications Server: IP IMS Sockets Guide*.

- To an implicit/explicit IMS transaction from the IMS transaction manager.

Refer to the information about DL/I status codes, return codes, and reason codes in the *IMS/ESA Diagnosis Guide and Reference*.

- To an implicit/explicit IMS transaction from the IMS database manager.

Refer to the information about DL/I status codes, return codes, and reason codes in the *IMS/ESA Diagnosis Guide and Reference*.

Where to find error message documentation

The following list refers to the appropriate error message documentation for all error messages expected in the IMS TCP/IP Services socket interface system.

- Error messages from the Listener are written to the SYSPRINT ddname data set. Refer to the information about the IMS Listener error messages in the *z/OS Communications Server: IP IMS Sockets Guide* for descriptions of the error messages in this data set.
- Error messages from TCP/IP are written to the SYSERROR and SYSDEBUG data sets. Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for descriptions of the error messages in these data sets.

Chapter 29. Diagnosing restartable VMCF/TNF problems

You can configure virtual machine communication facility (VMCF) and termination notification facility (TNF) in two different ways: as restartable subsystems or as nonrestartable subsystems. (For details on configuration, refer to the *z/OS Communications Server: IP Configuration Reference*.) If you choose restartable VMCF and TNF, you may encounter the following problems.

VMCF or TNF fail to initialize

If VMCF or TNF fail to initialize with an OC4 abend, there is probably an installation problem. Check the PPT entries for errors. Some levels of MVS do not flag PPT syntax errors properly.

Abends 0D5 and 0D6

If, after removing a user, the system crashes with abends 0D5 and 0D6, the application is probably still running and using VMCF. Users should not be removed from VMCF or TNF without first terminating the affected user.

No response to commands

If VMCF and TNF do not respond to commands, one or both of the nonrestartable versions of VMCF or TNF are still active. To get them to respond, follow these steps:

1. Stop all VMCF and TNF users.
2. Stop the subsystems using the commands `FORCE ARM VMCF` and `FORCE ARM TNF`.
3. Restart using `EZAZSSI`.

VMCF or TNF will not stop

If you are unable to stop VMCF or TNF, users probably still exist in the VMCF and TNF lists. Use the `F VMCF,DISPLAY,NAME=*` and the `F TNF,DISPLAY,NAME=*` commands to identify those users who are still active; then either cancel those users or remove them from the lists, using the `F VMCF,REMOVE` and the `F TNF,REMOVE` commands.

Chapter 30. Diagnosing problems with CICS

This chapter describes how to diagnose problems with the Customer Information Control System (CICS). CICS is an IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs.

For additional information that may be helpful in solving problems with CICS, refer to the following manuals:

- *z/OS Communications Server: IP CICS Sockets Guide*
- *CICS/ESA 5.2 Diagnosis Reference*
- *CICS/ESA 5.2 Problem Determination Guide*
- *CICS/ESA 5.2 Messages and Codes*
- *CICS/ESA 5.2 Data Areas*
- *CICS/ESA 5.2 Supplementary Data Areas*
- *z/OS MVS Diagnosis: Tools and Service Aids*
- *CICS/ESA 5.2 Operations Guide*

Diagnostic data

To diagnose problems with CICS, some or all of the following data may be required:

- Message logs
 - System log
 - Message log at the transient-data destination specified by ERRORTD
- CICS external-trace data set (auxtrace)

Note: To obtain the CICS auxtrace, set the CICS user master-trace flag, the system master-trace flag, and the AP level 1 standard trace component.

- Component trace
 - Engine
 - Physical file system (PFS)
 - Socket
 - Socket (SOCKAPI)
 - Transmission control protocol (TCP)
- Dumps
 - CICS transaction dump, if captured.
 - Supervisor Call (SVC) dump. SVC dumps are also known as *console dumps* or *system dumps*.

Note: For hangs and loops, request an SVC dump of CICS, TCP/IP, and the TCPIPDS1 data space.

- NETSTAT SOCKET output
- NETSTAT CONN output

Initialization problems

This section describes some problems you may encounter when attempting to initialize CICS.

CICS sockets interface not initialized

If the CICS sockets interface did not initialize, follow the steps below:

1. Issue the EZAO,START,CICS command, and then check that the interface initializes.
 - a. If the interface initializes, check that EZACIC20 is in the Program Load Table (DFHPLT).

Putting EZACIC20 into the PLT will allow the CICS Sockets Interface to initialize on CICS address startup. Refer to the *z/OS Communications Server: IP CICS Sockets Guide* for more information.
 - b. If EZACIC20 is defined in the DFHPLT, check the message logs for failures.
 - c. If there are no messages, start CICS with auxtrace active, and then request an SVC dump of CICS.
 - d. Call the Support Center.
2. Verify that the socket Resource Definition Online (RDO) definitions have been properly installed and that the correct data sets are in the STEPLIB and DFHRPL concatenations.

CICS listener not initialized

If the CICS listener did not initialize, follow the steps below:

1. Use the EZAC transaction to verify that the listener is defined in the configuration file.
2. In the configuration-file record for that listener, verify that IMMEDIATE is set to YES, and then verify that the correct APPLID and port number are specified.
3. Verify that the listener is properly defined in a CICS RDO group and that the RDO group is in the proper group list.
4. Check the message logs for failures.
 - a. If there are no messages, start CICS with auxtrace active, and then request an SVC dump of CICS.
 - b. Call the Support Center.
5. If an EZY1292I message was issued, investigate why the CICS sockets interface did not initialize. (See “CICS sockets interface not initialized”.)

No CICS sockets messages issued

If no CICS sockets messages (error or informational) were issued, verify that the correct CICS transient-data queue is specified in the ERRORTD field in the configuration record for the CICS region. A *region* is the CICS address space.

TCP/IP clients unable to connect

If TCP/IP clients are unable to connect, follow these steps:

1. Verify that the listener is active by logging on to CICS, and then issue a CEMT I TASK command. Make sure that the listener name appears in the task list.
2. Verify that the listener is listening on the correct port number by issuing a NETSTAT CONN command, and then check that the listener has the correct port in listen status. Verify that clients are trying to connect to this port and to the correct IP address.
3. Check the ERRORTD log and verify that the EZY1291I message has been issued. If it has not been issued, look for messages indicating a failure.

Child-server transactions not starting

Child-server transactions are transactions started by the listener. If child-server transactions are not starting, follow these steps:

1. Issue a CEMT I TRANSACTION command to verify that the transaction is installed. If it is not installed, a NOT FND message is displayed.
2. Issue a CEMT I PROGRAM command to verify that the child-server program is installed.
3. If the transaction or program is not installed, define it in the proper RDO group.
4. Check the message logs for failures.

CICS sockets application problems

This section describes some of the problems you may encounter with CICS sockets applications.

Hung CICS tasks

If CICS application tasks hang, follow these steps:

1. While a task is hung, request an SVC dump of CICS, TCP/IP, and the TCPIPDS1 data space.
2. If the problem can be recreated, recreate with CICS auxtrace and component trace turned on.
3. Issue a NETSTAT SOCKET command to determine if the task is waiting on a particular socket call to be posted. If it is waiting, you can issue the NETSTAT DROP command to terminate it.
4. If the application is hung while awaiting completion of a READ command, consider issuing a SELECT command prior to the READ command. The SELECT command returns either the number of sockets ready to be read or 0 (zero) if it times out.

Hung CICS region

If a CICS sockets application program using the Call Instruction API (EZASOKET) is erroneously link-edited without the EZACICAL stub, the entire CICS region may hang while waiting for socket calls to complete. Ensure that EZACICAL is explicitly link-edited with the application.

An EZASOKET call should generate a static call to the EZASOKET entry point within the EZACICAL stub. If the application is not compiled and link edited correctly, the EZASOKET call generates a dynamic call to program EZASOKET, which calls the socket API directly.

Errors on socket calls

If you receive errors on socket calls, note the ERRNO that is received, and then look it up in the section of the *z/OS Communications Server: IP CICS Sockets Guide* that describes return codes.

CICS shutdown hangs

If an EZY1342I message has been issued, there is a CICS task that has at least one socket open and that is not terminating. You can fix this problem by executing an immediate termination of the CICS sockets interface rather than a deferred termination. To execute an immediate termination, issue an EZAO,STOP,CICS command, and then specify YES at the IMMEDIATE prompt.

If you do not add EZACIC20 to the shutdown DFHPLT, CICS cannot terminate because the socket subtasks are still attached to the CICS region. To terminate CICS without EZACIC20, manually shut down the CICS sockets interface using the EZAO transaction.

CICS sockets control blocks

This section describes some problems you may encounter with the task interface element (TIE) and global work area (GWA). For information about the layout of GWA, TIE, and other control blocks, refer to the section in the *z/OS Communications Server: IP CICS Sockets Guide* that describes external data structures.

Task interface element

A TIE represents a CICS task that has issued at least one call to the CICS sockets API. You can locate TIEs in a dump of the CICS region by issuing the IPCS VERBX 'UEH=3' command, and searching for EZACIC01.TIE. The EZACIC01 prefix identifies it as a TIE for CICS sockets.

The UEH=3 output shows a CICS image of the TIE. The TCP/IP TIE is embedded within the CICS image of the TIE and starts at offset +X'80'.

Note: The UEH=3 output contains TIEs for other interfaces as well.

Global work area

The GWA is the main anchor point for the CICS sockets interface. It contains general status data, work areas, and pointers to other control-block chains. You can locate the GWA in a dump of the CICS region by issuing the IPCS command VERBX 'UEH=3', and searching for EZACIC01.GWA. The EZACIC01 prefix identifies it as the GWA for CICS sockets.

CICS trace

The CICS sockets task related user exit (TRUE), EZACIC01, issues CICS trace entries at the following four points of execution:

- When the TRUE receives a socket call from an application
- When the TRUE is passing the socket call to the subtask
- When the TRUE receives the response from the subtask
- When the TRUE is ready to return its response to the application

The trace point ID is AP 00C7. Trace records are self-explanatory. They show the type of call, the point of execution, the ERRNO, and the RETCODE.

Trace records can be written either to a CICS internal trace table or to its external-trace data set (auxtrace). To display the internal trace, follow these steps:

1. Request a dump of the CICS region using the RGN SDATA=(option 1,option 2...option n) parameter on a DUMP command. Examples of options are CSA, PSA, NVC, RGN, TRT, SQA, LSQA, LPA, and so on. For a complete list of options, refer to *z/OS MVS Diagnosis: Tools and Service Aids*.
2. Display the trace using the VERBX 'TR=2' IPCS command.

Note: CICS trace can also be directed to the GTF trace data set.

To display the auxtrace, follow the instructions for formatting auxtrace as documented in the *CICS/ESA 5.2 Operations Guide*.

Chapter 31. Diagnosing problems with Express Logon

The Express Logon feature in Communications Server for z/OS allows a user on a workstation, with a TN3270E client and an X.509 certificate, to log on to an SNA application without entering an ID or password.

This chapter describes how to diagnose problems with Express Logon for the z/OS V1R2 Communications Server Express Logon feature, including the Digital Certificate Access Server (DCAS). For complete information about Express Logon, refer to the User's Guide at <http://www.ibm.com/software/network/commserver/library/whitepapers/csos390.html>

For most situations in which the DCAS does not start, a message to the console is displayed. If the explanation in *z/OS Communications Server: IP and SNA Codes* does not help, you should turn on debugging and logging. You can specify debugging and logging as startup parameters from the z/OS UNIX shell or from the MVS console as a started procedure:

- If the DCAS is started from the z/OS UNIX shell, you can specify the following:

```
dcas -d <debugging_level> -l <logtype>
```

- If the DCAS is started from the MVS console, you can specify debugging and logging on the PARM statement after the final slash, as shown in the following example:

```
//DCAS    PROC  
//*  
//DCAS    EXEC PGM=EZADCDMN,REGION=4096K,TIME=NOLIMIT,  
// PARM='POSIX(ON) ALL31(ON) / -d 1 SYSLOGD'
```

The following optional parameters can be used with both DCAS UNIX commands and MVS started procedures:

-d or -D

Indicates debugging. The following levels apply:

- | | |
|----------|--|
| 1 | Specifies log error and warning messages. |
| 2 | Specifies log error, warning, and informational messages. |
| 3 | Specifies log error, warning, informational, and debug messages. |

The default level is 3.

-l or -L

Indicates logging to SYSLOGD or to a designated log file. If you do not specify this parameter, logging defaults to /tmp/dcas.log.

If you specify a debug level, but not logging, the DCAS attempts to open the default log file /tmp/dcas.log. If this fails, debugging is turned off.

For SYSLOGD, the DCAS uses the log facility local0.

For further aid in diagnosing errors, refer to the error logs of the TN3270E middle-tier servers. The HOD client security message panel should also be examined.

The following netstat commands, issued from the middle-tier server, are useful in determining connectivity problems between z/OS Communications Server and DCAS. For AIX®, the netstat command is:

```
netstat -an | grep port#
```

For CS/2, the netstat command is:

```
netstat -sn | grep port#
```

For NT, the netstat command is:

```
netstat -an | more port#
```

In the netstat commands, port# is the listening port of DCAS. The default DCAS port is 8990.

Analyzing start problems with the DCAS

When analyzing problems that occur when starting the DCAS, consider the following:

- The DCAS must run from an APF Authorized library.
- The DCAS uses z/OS Language Environment® C run-time services. Make sure that the LE C run-time library is compatible with the current level of z/OS Communications Server.
- The DCAS uses SSL cryptographic services run-time library. Verify that *hlq.SGSKLOAD* is accessible at run time. If certificates are authenticated using the X.500 host, SSL uses LDAP services to access the X.500 host. If running from the z/OS UNIX shell, verify that the LIBPATH environment variable includes */usr/lib*.
- The DCAS attempts to initialize SSL services. If you are using keyrings that reside in the HFS, verify that the KEYRING and STASHFILE keywords in the DCAS configuration file point to valid HFS file names. Names are case sensitive. If using keyrings that reside in RACF, verify that the SAFKEYRING keyword in the DCAS configuration file references a valid RACF keyring.
- The DCAS must be associated with a valid user ID using z/OS UNIX services. It must run with the POSIX(ON) C run-time option. Use the following RACF command:

```
ADDUSER dcasid DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
```

- If the DCAS is started as an MVS started procedure, verify that the following RACF commands have been issued:

```
RDEFINE STARTED DCAS.* STDATA(USER(dcasid))
RDEFINEOPERCMDS (MVS.SERVGR.DCAS) UACC(NONE)
PERMIT MVS.SERVGR.DCAS CLASS(OPERCMDS) ACCESS(CONTROL) ID(dcasid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```
- The DCAS uses the TCP/IP protocol to communicate with clients in the network. Verify that the z/OS Communications Server products VTAM and TCP/IP have been started and are active.

Analyzing client interface problems

When analyzing problems with client interfaces, consider the following:

- DCAS uses the TCP/IP protocol to communicate with its clients, the TN3270 middle-tier servers. Verify that the z/OS Communications Server products VTAM and TCP/IP have been started and are active. To verify network connectivity to a client, try pinging that client.
- The DCAS uses RACF services to obtain a user ID given a digital certificate.
 - Verify the certificate has been defined properly to RACF. Use the following commands:

```

SETOPTS CLASSACT(DIGTCERT)
SETOPTS RACLIST(DIGTCERT)    REFRESH
PERMIT IRR.DIGTCERT.function CLASS(FACILITY) ID(dcasid) ACCESS(CONTROL)
RACDCERT ID(userid)  ADD('certificate dataset name') TRUST

```

- Verify that the user ID associated with the DCAS has permission to access certificates. Use the following RACF commands:

```

SETOPTS CLASSACT(DIGTCERT)
SETOPTS RACLIST(DIGTCERT)    REFRESH
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(dcasid) ACCESS(CONTROL)

```

- The DCAS uses RACF services to obtain a PassTicket for an associated application ID. Verify that the RACF PTKTDATA profile for the application ID has been defined properly. The ID must match the ID specified on the workstation client. For HOD V5, this is the name specified in the Express Logon Application ID pop-up window. It may not be the same name specified on the USSMSG10. For applications such as TSO, specifying the application ID can be difficult since the profile name has special RACF considerations. Refer to the *z/OS Security Server RACF Security Administrator's Guide*.

Use these commands to verify the RACF PTKTDATA profile:

```

SETOPTS CLASSACT(PTKTDATA)
RDEFINE profile PTKTDATA SSIGNON()
SETOPTS RACLIST(PTKTDATA) REFRESH

```


Chapter 32. Diagnosing resolver problems

The resolver provides two kinds of tracing plus an IPCS subcommand to help analyze resolver problems in dumps. The resolver provides TRACE RESOLVER information that can be helpful in debugging problems an application program could have with using resolver facilities (for example, GetHostByName or GetHostByAddr). Component Trace is used for tracing the RESOLVER component (SYSTCPRE) for diagnosing resolver problems that cannot be isolated to one particular application. Use the IPCS RESOLVER subcommand to format and summarize resolver control blocks (see “RESOLVER” on page 267).

Refer to the *z/OS Communications Server: IP Configuration Reference* for additional information.

Steps for resolving the hostname

If a hostname fails to resolve, use the following information to help find the problem.

Before you begin: You need to know the exact hostname that failed to resolve and the environment in which the application was running (for example, TSO, UNIX, or batch).

1. Diagnosis why the hostname failed to resolve by pinging the hostname. Base your next course of action on the following conditions:

If ping for the hostname. . .	Then. . .	Solution
Succeeds, but another application fails when resolving the same hostname	The problem is with the resolver configuration for the application in the user's environment.	Use the Trace Resolver to solve the problem.
Fails, but the hostname is converted to an IP address	The resolution is successful but the host is not reachable or active.	See Figure 3 on page 26 to continue researching the problem.
Fails to convert the name to an IP address	The problem may be with the resolver configuration, searching local host files, or using DNS.	If you expect to resolve the hostname using DNS you should activate the Trace Resolver to solve the problem. Note: You can use the LOOKUP option in TCPIP.DATA to specify local searching before or instead of asking DNS.

2. Determine if the name or address being queried is known to DNS if you expect to resolve the hostname using DNS.

The following example will look for the name www.johndoe.com at IP address 1.2.3.4:

```
dig@1.2.3.4 www.johndoe.com -t any
```

The command should return all resource records of any type from the DNS at 1.2.3.4 for www.johndoe.com. For more information about dig, see *z/OS Communications Server: IP System Administrator's Commands*.

3. If dig does not return all resource records, base your next course of action on the following conditions:

If dig. . .	Then . . .	Solution
Fails because it cannot contact DNS	You need to check your link to the DNS IP address.	See Figure 3 on page 26 to continue researching the problem.
Fails because DNS reports that the resource was not found	www.johndoe.com is not a resource record known to DNS.	See the DNS administrator to add the name. As a temporary work around, you might want to add the name to a local host file that the Resolver searches. Refer to <i>z/OS Communications Server: IP Configuration Guide</i> for information about local host files.
Succeeds	The problem in resolving the hostname using ping or another application may be in configuring the resolver.	The dig command bypasses the Resolver search orders, local host files, and domain names appended by the Resolver. The best way to check the configuration is to start the Trace Resolver. It is important to use the Trace Resolver in the environment where the application is failing because the application may be using a different TCPIP.DATA file, environment variables, or search order than the environment where the dig command was issued.

You know you are done when the application that previously failed to resolve the hostname can now resolve it.

TRACE RESOLVER

The Trace Resolver tells what the Resolver looked for (the Questions) and where it looked (name servers' IP addresses or local host file names). Check the following in the trace output:

- Fix or check any problems reported at the top of the trace. These are errors in the resolver data sets.
- Are the data sets being used by the resolver the ones you expected? If not, see the search orders for data sets in the *z/OS Communications Server: IP Configuration Guide*.
- Check the TCPIP.DATA parameter values, especially Search, NameServer, and NsPortAddr. TCPIP.DATA parameters are explained in *z/OS Communications Server: IP Configuration Reference*.
- Check the questions posed by the Resolver to DNS or in searching the local host files. Are these the queries you expected?
- Look for errors or failures in the trace.
- Did DNS respond (if you expected it to)? If not, see if DNS is active at the IP address you specified for NameServer and what port it is listening on. Also DNS logs may be helpful. Ask the DNS administrator for help.
- Some common misunderstandings:

- If the queried name server returns NXDOMAIN, the resolver does not continue to the next name server in the list. NXDOMAIN means the domain does not exist according to that name server.
- The resolver only appends the specific names listed in the Search (or Domain) parameter. It does not attempt shorter versions of these. For example, if you look for "johndoe" and your search list has "anywhere.usa.com", the resolver will look for "johndoe.anywhere.usa.com" and "johndoe" (the order depends on the value of option ndots). The Resolver will not look for "johndoe.anywhere" or "johndoe.anywhere.usa" or "johndoe.usa.com" or "johndoe.com".

The TRACE RESOLVER output includes:

- The full TCPIP.DATA data set name.
- An indication of whether or not the DefaultTCPIPData and GlobalTCPIPData parameters were specified.
- The value and origin of all the parameters being used by the Resolver. The origin would be either the file name or the data set name where the value was picked up or default.
- The IP address for each name server, but not its name.
- An indication as to which API is invoking the resolver services
- Calls and parameters passed to and returned from the resolver API functions.
- Queries to and response from DNS.
- Searches and results of local host files

The TRACE RESOLVER for an application or user will be activated and written to a file specified in one of the following ways:

- In an environment variable or in a DD statement
- Implied by the TRACE RESOLVER or OPTIONS DEBUG statement
- Implied by the application setting the debug option in the _res structure

Note: An LE batch job can specify both an environment variable and a DD statement.

A member of a PDS is not supported. The following search order is used in all cases whether the resolver is called from the LE or the TSO environment.

1. Environment variable RESOLVER_TRACE. The value can be either the constant stdout, a data set name, or an HFS file name.

To direct the trace resolver output in the z/OS UNIX shell to standardout, enter the following command:

```
export RESOLVER_TRACE=stdout
```

To disable the trace resolver output, issue the following command:

```
set -A RESOLVER_TRACE
```

Note: This step will fail if the data set or file is not available or has been allocated elsewhere.

2. Any data set that is allocated to the new DD-name SYSTCPT. This option is not recommended for processes that use the fork() command. This allocation will not be available to the child process that is forked because DD allocations for the parent process are not inherited by the child. The only exception to this rule is a STEPLIB allocation.

For the batch job, specify the DD card in the job procedure JCL:

```
//SYSTCPT DD SYSOUT=*
or
//SYSTCPT DD DISP=SHR,DSN=USERID.TRACE.RESOLVER
```

The output can be sent to the TSO terminal by using the TSO command:

```
ALLOC DD(SYSTCPT) DA(*)
```

When directing Trace Resolver output to a TSO terminal the screen size can be defined to have only 80 columns. To disable the Trace Resolver output use the TSO command:

```
FREE DD(SYSTCPT)
```

3. If the TRACE RESOLVER statement is in TCPIP.DATA, the trace output will go to stdout for UNIX applications and to SYSPRINT for TSO applications.

Note: If all applications share the same TCPIP.DATA, all the applications and users will be traced. This is not recommended.

4. If the OPTIONS DEBUG statement is in TCPIP.DATA, the trace output will go to stdout for UNIX applications and to SYSPRINT for TSO applications. If SYSPRINT is used, it is recommended that it be directed to SYSOUT=*. If that cannot be used, then use either the SYSTCPT or RESOLVER_TRACE support to direct the trace output to the desired location.

Note: If all applications share the same TCPIP.DATA, all the applications and users will be traced. This is not recommended.

5. If the application sets the DEBUG bit in the options field of the _res structure, the trace output will be directed to stdout for UNIX applications and to SYSPRINT for TSO applications.

The Resolver will search for the above values at the time of res_init() processing and when the MODIFY REFRESH command is issued. The presence of the environment variable, DD statement, TRACE RESOLVER or OPTIONS DEBUG statement, or DEBUG option will cause the trace to be activated. If the selected trace file is not available or has been allocated exclusively elsewhere, a console message will be printed to that effect and the Resolver Trace will not execute.

The resolver begins tracing when a command is issued such as GetHostByName. The various methods of specifying that trace is active will be checked first. If tracing is active and the trace data set is writeable, the remaining res_init processing will be traced.

The DCB characteristics of the Trace Resolver data set must be specified because the data will be truncated if the record length is too short. The LRECL can be between 80 and 256 and the RECFM will be FIXED. For an LRECL of 128 or larger the last 6 positions will be the storage address of the MVS TCB that issued the resolver call.

Note: If the trace resolver output gets filled up, the results depend on where the output is directed. The trace resolver is running in the application's address space. If the output is directed to a JES spool file and JES is running out of space, the application waits. If the output is directed to a data set and it is filled up, the application receives an x37 ABEND. If the output is directed to a tape, the application will have to wait up to 5 minutes; if a tape is not available, it will ABEND with x37.

The following is an example showing the setup files used, the command used to invoke the trace, and the trace resolver output:

- **Setup files used for trace resolver:**

- **Resolver Procedure:**

```
//RESOLVER PROC PARMS='CTRACE(CTIRESFL)'
//*
//EZBREINI EXEC PGM=EZBREINI,REGION=0M,TIME=1440,PARM=&PARMS
//*
//SETUP DD DSN=TPOUSER.RESOLVER.SETUP.DATA,DISP=SHR,FREE=CLOSE
```

- **Setup File TPOUSER.RESOLVER.SETUP.DATA contains:**

```
;
DEFAULTTCPIPDATA('TPOUSER.RESOLVER.DEFAULT.DATA')
;
; GLOBALTCPIPDATA(/ETC/TCIPGLOBAL.DATA)
#
GLOBALTCPIPDATA('SYS1.TCPPARMS(RESGLOBL)')
```

- **Global TCPIP.DATA file SYS1.TCPPARMS(RESGLOBL) contains:**

```
# Note that DOMAIN is ignored because SEARCH is mutually exclusive
# and SEARCH appears after DOMAIN.
Domain abcxyz
; Note that SEARCH can be specified on multiple lines.
SEARCH tcp.raleigh.ibm.com raleigh.ibm.com
SEARCh ibm.com com uk
SEARCh gov
1a Search mil
SORTLIST 0.0.19.0/0.0.255.0 0.0.18.99/0.0.255.255 0.42.17.0/0.255.255.0
SORTLIST 129.42.16.0/255.255.255.0
1b Sortlist 9.0.0.0
NSinterAddr 9.67.128.82 ; Buzz
NameServer 9.67.128.255 ; not a server
NSportAddr 53
2 ;ResolveVia UDP
ResolverTimeout 3
ResolverUdpRetries 1
1c loadbcstables unknown
loadbcstables big5
3 MVS026: Hostname MVS026
```

- **Default TCPIP.DATA file TPOUSER.RESOLVER.DEFAULT.DATA contains:**

```
; TRACE RESOLVER
DatasetPrefix USER1
TcpiJobname TCPCS3
Hostname VIC097
; trace c sockets
; alwayswto no
; messagecase whoknows
; loadbcstables tbd
```

Note: For this example, this file exists but is not used in the procedure for obtaining this example trace resolver output.

- **Local TCPIP.DATA file USER55.TCPIP.DATA contains:**

```
; trace resolver
DATASETPREFIX USER55
# If an option is coded multiple times but can only have 1 value,
# the last occurrence is used.
TCPIPjobname TCPCS2
TCPIPjobname TCPCS
3 HostName MVS000
DomainOrigin edu
;
NameServer 127.0.0.1 ; loopback
#
2 ResolveVia TCP
```

```

ResolverTimeout 22
1d alwayswto xyz
messagecase mixed
loaddbcsables schinese

```

• **TSO commands issued to obtain the trace (gethostbyname):**

```

alloc dd(systcpt) dsn(traceres) reuse
4 invoke a REXX application which issues gethostbyname for www.ibm.com

```

• **Trace Resolver output in USER55.TRACERES contains (gethostbyname):**

```

5 Resolver Trace Initialization Complete -> 2001/04/26 13:09:37.509773
1a res_init Skipped option(s) on line 8: SYS1.TCPPARMS(RESGLOBL)
1b res_init Skipped option(s) on line 11: SYS1.TCPPARMS(RESGLOBL)
1c res_init Parse error on line 18: SYS1.TCPPARMS(RESGLOBL)
1d res_init Parse error on line 14: USER55.TCPIP.DATA

```

```

6 res_init Resolver values:
Global Tcp/Ip Dataset = SYS1.TCPPARMS(RESGLOBL)
Default Tcp/Ip Dataset = TPOUSER.RESOLVER.DEFAULT.DATA
Local Tcp/Ip Dataset = USER55.TCPIP.DATA
Translation Table = Default
UserId/JobName = USER55
Caller API = TCP/IP REXX Sockets
(L) DataSetPrefix = USER55

```

```

3 (G) HostName = MVS026
(L) TcpIpJobName = TCPCS
(G) Search = tcp.raleigh.ibm.com
           raleigh.ibm.com
           ibm.com
           com
           uk
           gov
(G) SortList = 0.0.19.0/0.0.255.0
              0.0.18.99/0.0.255.255
              0.42.17.0/0.255.255.0
              129.42.16.0/255.255.255.0
(G) NameServer = 9.67.128.82
                9.67.128.255
(G) NsPortAddr = 53 (G) ResolverTimeout = 3
(*) ResolveVia = UDP (G) ResolverUdpRetries = 1
(*) Options NDots = 1
(*) SockNoTestStor
(*) AlwaysWto = NO (L) MessageCase = MIXED
(G) LoadDbcsTable = BIG5
11 (*) LookUp = DNS LOCAL
res_init Succeeded

```

```

4 GetHostByName Resolving Name: WWW.IBM.COM
res_search(WWW.IBM.COM, C_IN, T_A)
res_search Host Alias Search found no alias
res_querydomain(WWW.IBM.COM., , C_IN, T_A)
res_querydomain resolving name: WWW.IBM.COM.
res_query(WWW.IBM.COM., C_IN, T_A)
res_mkquery(QUERY, WWW.IBM.COM., C_IN, T_A)

```

```

7 res_mkquery created message:
* * * * * Beginning of Message * * * * *
Query Id: 62981
Flags: 00000001 00000000
Flags set: recurDes
OpCode: QUERY
Response Code: NOERROR

```

```

Number of Question RRs: 1
Question 1:
WWW.IBM.COM
Type (0X0001) T_A Class (0X0001) C_IN

```

```

Number of Answer RRs: 0
Number of Authority RRs: 0
Number of Additional RRs: 0
* * * * * End of Message * * * * *
8 res_send Sending query to Name Server 9.67.128.82
  BPX1SOC: RetVal = 0, RC = 0, Reason = 0x00000000
  BPX1IOC: RetVal = 0, RC = 0, Reason = 0x00000000
  BPX1ST0: RetVal = 29, RC = 0, Reason = 0x00000000
  BPX1SEL: RetVal = 1, RC = 0, Reason = 0x00000000
  BPX1RCV: RetVal = 139, RC = 0, Reason = 0x00000000
  UDP Data Length: 139
9 res_send received data via UDP. Message received:
  * * * * * Beginning of Message * * * * *
  Query Id:          62981
  Flags:             10000001 10000000
  Flags set:         resp recurDes recurAvl
  OpCode:            QUERY
  Response Code:     NOERROR

  Number of Question RRs: 1
  Question 1:
  WWW.IBM.COM
  Type (0X0001) T_A Class (0X0001) C_IN

  Number of Answer RRs: 4
  Answer 1:
  WWW.IBM.COM
  Type (0X0001) T_A Class (0X0001) C_IN
  TTL: 0093 (0 days, 0 hours, 1 minutes, 33 seconds)
  129.42.16.99
  Answer 2:
  WWW.IBM.COM
  Type (0X0001) T_A Class (0X0001) C_IN
  TTL: 0093 (0 days, 0 hours, 1 minutes, 33 seconds)
  129.42.17.99
  Answer 3:
  WWW.IBM.COM
  Type (0X0001) T_A Class (0X0001) C_IN
  TTL: 0093 (0 days, 0 hours, 1 minutes, 33 seconds)
  129.42.18.99
  Answer 4:
  WWW.IBM.COM
  Type (0X0001) T_A Class (0X0001) C_IN
  TTL: 0093 (0 days, 0 hours, 1 minutes, 33 seconds)
  129.42.19.99

  Number of Authority RRs: 2
  Authority 1:
  WWW.IBM.COM
  Type (0X0002) T_NS Class (0X0001) C_IN
  TTL: 33827 (0 days, 9 hours, 23 minutes, 47 seconds)
  ns.nyc.ibm.com
  Authority 2:
  WWW.IBM.COM
  Type (0X0002) T_NS Class (0X0001) C_IN
  TTL: 33827 (0 days, 9 hours, 23 minutes, 47 seconds)
  ns2.nyc.ibm.com

  Number of Additional RRs: 0
  * * * * * End of Message * * * * *
  BPX1CLO: RetVal = 0, RC = 0, Reason = 0x00000000
  res_send Succeeded
  res_query Succeeded
  res_querydomain Succeeded
  res_search Succeeded
10 GetHostByName Succeeded: IP Address(es) found:

```

```

IP Address(1) is 129.42.19.99
IP Address(2) is 129.42.18.99
IP Address(3) is 129.42.17.99
IP Address(4) is 129.42.16.99

```

- **TSO commands issued to obtain the trace (getaddrinfo):**

```

alloc dd(systcpt) dsn(traceres) reuse
4 ping cs390-2e

```

```

Ping CS V1R4: Pinging host CS390-2E.tcp.raleigh.ibm.com
at IPv6 address fec9:c2d4::9:67:115:7
sendto(): EDC8130I Host cannot be reached.

```

- **Trace Resolver output in USER55.TRACERES contains (getaddrinfo):**

```

5 Resolver Trace Initialization Complete -> 2002/02/08 11:40:37.237177
1a res_init Skipped option(s) on line      8: SYS1.TCPPARMS(RESGLOBL)
1b res_init Skipped option(s) on line     11: SYS1.TCPPARMS(RESGLOBL)
1c res_init Parse error on line          18: SYS1.TCPPARMS(RESGLOBL)
1d res_init Parse error on line          14: USER55.TCPIP.DATA

6 res_init Resolver values:
Global Tcp/Ip Dataset = SYS1.TCPPARMS(RESGLOBL)
Default Tcp/Ip Dataset = TPOUSER.RESOLVER.DEFAULT.DATA
Local Tcp/Ip Dataset = USER55.TCPIP.DATA
Translation Table = Default
UserId/JobName = USER55
Caller API = TCP/IP Sockets Extended
(L) DataSetPrefix = USER55
3 (G) HostName = MVS026
(L) TcpIpJobName = TCPCS
(G) Search = tcp.raleigh.ibm.com
           raleigh.ibm.com
           ibm.com
           com
           uk
           gov
(G) SortList = 0.0.19.0/0.0.255.0
              0.0.18.99/0.0.255.255
              0.42.17.0/0.255.255.0
              129.42.16.0/255.255.255.0
(G) NameServer = 9.67.128.82
               9.67.128.255
(G) NsPortAddr = 53
2 (*) ResolveVia = UDP
(*) Options NDots = 1
(*) SockNoTestStor
(*) AlwaysWto = NO
(G) LoadDbcsTable = BIG5
11 (*) LookUp = DNS LOCAL
res_init Succeeded
12 GetAddrinfo Invoked with following inputs:
Host Name: CS390-2E
No Service operand specified
Hints parameter supplied with settings:
ai_family = 0, ai_flags = 0x00000062
ai_protocol = 0, ai_socktype = 0
13 GetAddrInfo Opening Socket for IOCTls
BPX1SOC: RetVal = 0, RC = 0, Reason = 0x00000000
GetAddrInfo Opened Socket 0x00000000
14 GetAddrInfo Both IPv4 and IPv6 Interfaces Exist
GetAddrInfo Host Alias Search found no alias
res_querydomain(CS390-2E, tcp.raleigh.ibm.com, C_IN, T_AAAA)
res_querydomain resolving name: CS390-2E.tcp.raleigh.ibm.com

```

```

15 res_query(CS390-2E.tcp.raleigh.ibm.com, C_IN, T_AAAA)
res_mkquery(QUERY, CS390-2E.tcp.raleigh.ibm.com, C_IN, T_AAAA)
res_mkquery created message:
* * * * * Beginning of Message * * * * *
Query Id:                63243
Flags:                   00000001 00000000
Flags set:               recurDes
OpCode:                  QUERY
Response Code:           NOERROR

Number of Question RRs:  1
Question 1:
CS390-2E.tcp.raleigh.ibm.com
Type (0X001C) T_AAAA Class (0X0001) C_IN

Number of Answer RRs:    0
Number of Authority RRs: 0
Number of Additional RRs: 0
* * * * * End of Message * * * * *
res_send Sending query to Name Server 9.67.128.82
BPX1STO: RetVal = 46, RC = 0, Reason = 0x00000000
BPX1SEL: RetVal = 1, RC = 0, Reason = 0x00000000
BPX1RCV: RetVal = 109, RC = 0, Reason = 0x00000000
UDP Data Length: 109

res_send received data via UDP. Message received:
* * * * * Beginning of Message * * * * *
Query Id:                63243
Flags:                   10000101 10000000
Flags set:               resp auth recurDes recurAvl
OpCode:                  QUERY
Response Code:           NOERROR

Number of Question RRs:  1
Question 1:
CS390-2E.tcp.raleigh.ibm.com
Type (0X001C) T_AAAA Class (0X0001) C_IN

Number of Answer RRs:    1
Answer 1:
CS390-2E.tcp.raleigh.ibm.com
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 86400 (1 days, 0 hours, 0 minutes, 0 seconds)
FEC9:C2D4::9:67:115:7

Number of Authority RRs: 1
Authority 1:
tcp.raleigh.ibm.com
Type (0X0002) T_NS Class (0X0001) C_IN
TTL: 86400 (1 days, 0 hours, 0 minutes, 0 seconds)
buzz.tcp.raleigh.ibm.com

Number of Additional RRs: 1
Additional 1:
buzz.tcp.raleigh.ibm.com
Type (0X0001) T_A Class (0X0001) C_IN
TTL: 86400 (1 days, 0 hours, 0 minutes, 0 seconds)
9.67.128.82
* * * * * End of Message * * * * *
res_send Succeeded

res_query Succeeded
res_querydomain Succeeded

```

```

res_querydomain(CS390-2E, tcp.raleigh.ibm.com, C_IN, T_A)
res_querydomain resolving name: CS390-2E.tcp.raleigh.ibm.com
16 res_query(CS390-2E.tcp.raleigh.ibm.com, C_IN, T_A)
res_mkquery(QUERY, CS390-2E.tcp.raleigh.ibm.com, C_IN, T_A)
res_mkquery created message:
* * * * * Beginning of Message * * * * *
Query Id:                63244
Flags:                   00000001 00000000
Flags set:               recurDes
OpCode:                 QUERY
Response Code:          NOERROR

Number of Question RRs:  1
Question 1:
CS390-2E.tcp.raleigh.ibm.com
Type (0X0001) T_A Class (0X0001) C_IN

Number of Answer RRs:   0
Number of Authority RRs: 0
Number of Additional RRs: 0
* * * * * End of Message * * * * *
res_send Sending query to Name Server 9.67.128.82
BPX1ST0: RetVal = 46, RC = 0, Reason = 0x00000000
BPX1SEL: RetVal = 1, RC = 0, Reason = 0x00000000
BPX1RCV: RetVal = 97, RC = 0, Reason = 0x00000000
UDP Data Length: 97
res_send received data via UDP. Message received:
* * * * * Beginning of Message * * * * *
Query Id:                63244
Flags:                   10000101 10000000
Flags set:               resp auth recurDes recurAvl
OpCode:                 QUERY
Response Code:          NOERROR

Number of Question RRs:  1
Question 1:
CS390-2E.tcp.raleigh.ibm.com
Type (0X0001) T_A Class (0X0001) C_IN

Number of Answer RRs:   1
Answer 1:
CS390-2E.tcp.raleigh.ibm.com
Type (0X0001) T_A Class (0X0001) C_IN
TTL: 3600 (0 days, 1 hours, 0 minutes, 0 seconds)
9.67.115.7

Number of Authority RRs: 1
Authority 1:
tcp.raleigh.ibm.com
Type (0X0002) T_NS Class (0X0001) C_IN
TTL: 86400 (1 days, 0 hours, 0 minutes, 0 seconds)
buzz.tcp.raleigh.ibm.com

Number of Additional RRs: 1
Additional 1:
buzz.tcp.raleigh.ibm.com
Type (0X0001) T_A Class (0X0001) C_IN
TTL: 86400 (1 days, 0 hours, 0 minutes, 0 seconds)
9.67.128.82
* * * * * End of Message * * * * *
res_send Succeeded
res_query Succeeded

```

```

res_querydomain Succeeded
17 GetAddrInfo Returning Zero as Port Number
GetAddrInfo Built 2 Addrinfos
13 GetAddrInfo Closing IOCTL Socket 0x00000000
BPX1CLO: RetVal = 0, RC = 0, Reason = 0x00000000
18 GetAddrInfo Succeeded: IP Address(es) found:
    IP Address(1) is FEC9:C2D4::9:67:115:7
    IP Address(2) is 9.67.115.7
*****
FreeAddrInfo Called to free addrinfo structures
FreeAddrInfo Succeeded, Freed 2 Addrinfos
*****

```

The following describes highlighted numbered areas of the example setup files and example trace resolver output.

1

Errors deliberately entered into this example to show action taken.

a

Line 8 in the global file has 7 search values - the maximum is 6. The seventh one is ignored.

b

Line 11 in the global file has 5 sortlist values - the maximum is 4. The fifth one is ignored.

c

Line 18 in the global file has a value for LOADDBCSTABLES that is not valid. The value is ignored.

d

Line 14 in the local file has a value for ALWAYSWTO that is not valid. The value is ignored and the default is used.

2

ResolveVia is UDP even though the local file had ResolveVia TCP. UDP is used because GLOBALTCPIPDATA is being used. If a global file is used then all resolver related TCPIP.DATA statements must be specified in it. If the resolver statements are not specified then default values will be assigned. In this example, resolver statements are not specified as shown by ResolveVia in the global file being commented out.

3

A local file cannot override the global file for any value. The global file specifies the hostname, therefore the local file value of MVS000 will not override the global value of MVS026.

4

ping calls GetHostByName. The flow through the resolver API calls shows the parameters being passed.

5

Trace output reports the date and time the command was issued.

6

The res_init() resolver initialization values are reported. These are the values actually being used by the resolver, with an indication of the origin of the value. The indicators are:

* Default value

D Default file (not used if the local file is found)

- E** Environment variable
- G** Global file
- L** Local file

7 res_mkquery creates a DNS message (from Beginning of Message to End of Message). The message is interpreted, and flags and codes are spelled out.

8 res_send sends the query to the name server. The res_send function calls several z/OS UNIX functions; the indentation of the lines following res_send indicate res_send was the caller.

9 res_send receives a message from DNS. Note that there are 4 IP addresses for this name (4 answers).

10 GetHostByName function reports success, listing the IP addresses returned. Note the order of the addresses matches the sortlist specification.

11 LookUp specifies the order in which the DNS and the local host file are to be used for name resolution. It can be:

- LookUp = DNS LOCAL (DNS search first)
- LookUp = LOCAL DNS (Local host file search first)
- LookUp = DNS (only DNS search)
- LookUp = LOCAL (only Local host file search)

12 Ping calls GetAddrinfo. The flow through the resolver API calls shows the parameters being passed.

- ai_family = 0 means that AF_UNSPEC is specified
- ai_flags = x'00000062' means that AI_CANNONNAMEOK, AI_ALL, and AI_ADDRCONFIG are specified
- ai_protocol = 0 and ai_socktype = 0 means that protocol and socktype are not specified

Please refer to *z/OS Communications Server: IP Application Programming Interface Guide* for more information about input values of getaddrinfo.

13 In order to honor the setting of ai_ADDRCONFIG, the Resolver must query the stacks to determine whether IPv6 or IPv4 interfaces exist (the results of the query are shown in message **14**). A socket, separate from the one used to send DNS queries, is opened for communicating with the stacks.

14 The resolver detected that the system can handle both IPv4 and IPv6 addresses.

15 Because the system can handle both IPv4 and IPv6, and ai_ALL is specified, the resolver sends the IPv6 query (T_AAAA) for IPv6 to DNS first. For an explanation of how resolver decides to send an IPv6 or IPv4 query to DNS, refer to the *z/OS Communications Server: IPv6 Network and Application Design Guide*.

16

The resolver sends the IPv4 query (T_A) to DNS second. For an explanation of how resolver decides to send an IPv6 or IPv4 query to DNS, refer to the *z/OS Communications Server: IPv6 Network and Application Design Guide*.

17

Because no Service operand was passed as input to Getaddrinfo, there is no service resolution to perform, so any sockaddr returned will have a port number=0.

18

Prior to returning resolved addresses to the application, the resolver will sort all addresses so that the most preferable is the first in the address chain. Refer to the *z/OS Communications Server: IPv6 Network and Application Design Guide* for more information.

Notes:

1. If any errors occurred, refer to *z/OS Communications Server: IP and SNA Codes*.
2. In a multitasking environment, if the LRECL of the trace resolver output is at least 128 characters, the TCB address appears at the end of each line. The TCB address can be useful in determining the origin of the resolver trace statement.

CTRACE — RESOLVER

Component Trace (CTRACE) is used for the RESOLVER component (SYSTCPRE) to collect debug information. Currently the TRACE RESOLVER traces information on a per-application basis and directs the output to a unique file for each application. The CTRACE will show resolver actions for all applications (although it may be filtered).

The CTRACE support allows for JOBNAME, ASID filtering, or both. The trace buffer will be located in the Resolver private storage. The trace buffer minimum size will be 128K, maximum 128M, default 16M. Trace records can optionally be written to an external writer.

The Resolver CTRACE initialization PARMLIB member can be specified at Resolver start time. Using the sample Resolver procedure shipped with the product, enter the following console command:

```
S RESOLVER,PARMS='CTRACE(CTIRESxx)'
```

where xx is the suffix of the CTIRESxx PARMLIB member to be used. To customize the parameters used to initialize the trace, you can update the SYS1.PARMLIB member CTIRES00.

Note: In addition to specifying the trace options, you can also change the Resolver trace buffer size. The buffer size can be changed only at Resolver initialization.

If the CTIRES00 member is not found when starting the Resolver, the following message is issued:

```
IEEE538I CTIRES00 MEMBER NOT FOUND in SYS1.PARMLIB
```

When this occurs, the Resolver component trace is started with a buffer size of 16MB and the MINIMUM tracing option.

After Resolver initialization, you must use the TRACE CT command to change the component trace options (see Chapter 5, “TCP/IP services traces and IPCS support” on page 53). Each time a new component trace is initialized, all prior trace options are turned off and the new options are put into effect.

Trace options:

ALL

All options.

MINIMUM

The minimum set of options traces exceptions, Resolver initialization and termination, Resolver CTRACE changes, and Resolver operator messages.

Following is the sample PARMLIB member.

```

/*****
/*
/* TCP/IP for MVS
/* SMP/E Distribution Name: CTIRES00
/*
/* PART Name: CTIRES00
/*
/*
/* Copyright:
/* Licensed Materials - Property of IBM
/* 5694-A01
/* (C) Copyright IBM Corp. 2001
/*
/*
/* Status: CSV1R4
/*
/*
/* DESCRIPTION = This parmlib member causes component trace for
/* the TCP/IP provided Resolver to be initialized
/* with a trace buffer size of 16M
/*
/* This parmlib member only lists those TRACEOPTS
/* values specific to the TCP/IP Resolver. For a
/* complete list of TRACEOPTS keywords and their
/* values see:
/* OS/390 MVS INITIALIZATION AND TUNING REFERENCE.
/*
/*
/* Change Activity:
/*
/* $PARMS(CTIRES00),COMP(RES ),PROD(TCPIP ): Resolver Component Trace*/
/* SYS1.PARMLIB member
/*
/* CHANGE-ACTIVITY =
/* CFD List:
/*
/* Flag Reason Release Date Origin Description
/* -----
/* $L0= D312.22 CSV1R2 000324 AMITRANO: Resolver DCR
/* $L1= MV22533 D312.22 001211 McGeachy: Clarify storage origin
/* $L2= MV22673 D312.22 010116 McGeachy: Change options
/*
/*
*****/
TRACEOPTS
/* -----
/* Optionally start external writer in this file (use both
/* WTRSTART and WTR with same wtr_procedure)
/* -----
/* WTRSTART(wtr_procedure)
/* -----

```

```

/*  ON OR OFF: PICK 1                                     */
/* ----- */
/*      ON */
/*      OFF */
/* ----- */
/*  BUFSIZE: A VALUE IN RANGE 128K TO 128M                */
/*      CTRACE buffers reside in Resolver Private storage */
/*      which is in the regions address space.             */
/* ----- */
/*      BUFSIZE(16M) */
/*      JOBNAME(jobname1,...) */
/*      ASID(Asid1,...) */
/*      WTR(wtr_procedure) */
/* ----- */
/*  OPTIONS: NAMES OF FUNCTIONS TO BE TRACED, OR "ALL"    */
/* ----- */
/*      OPTIONS( */
/*          'ALL' */
/*          , 'MINIMUM' */
/*      ) */

```

When formatting the Resolver trace, use the CTRACE command. See Chapter 5, “TCP/IP services traces and IPCS support” on page 53 for the syntax for formatting a CTRACE. For the Resolver, the following formatting OPTIONS are available:

ASCII

Resolver trace data is displayed with ASCII translation only. The default is EBCDIC.

BOTH

Resolver trace data is displayed with both EBCDIC and ASCII translations. Each line of formatted data contains the offset, the hexadecimal display, the EBCDIC translation, then the ASCII translation. The default is EBCDIC.

EBCDIC

Resolver trace data is displayed with EBCDIC translation only. This is the default.

HEX

Resolver trace data is displayed only in hexadecimal (no ASCII or EBCDIC translation). The default is EBCDIC.

Note: If the formatted CTRACE display wraps on the screen, use the IPCS PROFILE LINESIZE(*nn*) command, where *nn* is the largest number of characters that will display on one line.

Chapter 33. Diagnosing Simple Network Time Protocol (SNTP) problems

The Simple Network Time Protocol (SNTP) is a protocol for synchronizing clocks across a WAN or LAN through a specific formatted message. The Simple Network Time Protocol Daemon (SNTPD) is a TCP/IP daemon that is used to synchronize time between a client and a server.

This chapter describes how to diagnose problems with SNTP.

Activating the SNTPD debug trace

To activate the SNTPD debug trace, specify the `-d` or `-df` option when starting SNTPD via z/OS UNIX shell or the TSO procedure.

- With the `-d` option, activity logging and debugging messages are written to stdout.
- With the `-df` option, a path name and file name must be specified; activity logging and debugging messages are written to this file.

Abends

An abend during SNTPD processing should result in messages and error related information being sent to the system console. A dump of the error will be needed unless the symptoms already match a known problem.

Steps for stopping SNTPD

Before you issue the kill command: you must determine the PID (process ID) of SNTPD.

Perform the following steps to stop the process ID of SNTPD.

1. To find the PID, use one of the following methods:
 - Use `D OMVS,U=USERID`. (This is the USERID that started ANTPD from the shell).
 - Use the `ps-ef` command from the shell.
 - Write down the PID when you start SNTPD.
2. From a z/OS UNIX shell superuser ID, issue the kill command to the process ID (PID) associated with SNTPD.

You know you are finished when the following message appears: EZZ9601I SNTP server is terminating.

Sample SNTPD debug output

Refer to *z/OS MVS Diagnosis: Procedures* or see Chapter 3, “Diagnosing abends, loops, and hangs” on page 21, for information about debugging dumps produced during SNTP processing.

```
Tue Apr 2 15:26:14 2002 SNTP enabled options: Opening debugging file /tmp/bc6.log
(Multicast: every 120 seconds) (PID FILE: /etc/sntpd.pid) (DEBUG FILE: /tmp/bc6.log
Tue Apr 2 15:26:14 2002 Writing PID to file /etc/sntpd.pid
```

```

| Tue Apr 2 15:26:14 2002 EZZ9602I SNTP server initializing
| Tue Apr 2 15:26:14 2002 Initializing signal handling
| Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGINT
| Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGTERM
| Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGABND
| Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGABRT
| Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGQUIT
| Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGHUP
| Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGTTOU
| Tue Apr 2 15:26:14 2002 Initializing MVS command handling
| Tue Apr 2 15:26:14 2002 Initializing pthread for MVS command
| Tue Apr 2 15:26:14 2002 Initializing UDP socket(s)
| Tue Apr 2 15:26:15 2002 SNTP port was set to 123
| Tue Apr 2 15:26:15 2002 Bound to address: 9.67.2.1
| Tue Apr 2 15:26:15 2002 Bound to address: 9.67.115.15
| Tue Apr 2 15:26:15 2002 Bound to address: 9.67.2.2
| Tue Apr 2 15:26:15 2002 Bound to address: 0.0.0.0
| Tue Apr 2 15:26:15 2002 Initializing pthread for multicast/broadcast
| Tue Apr 2 15:26:15 2002 Initializing pthread for unicast
| Tue Apr 2 15:26:15 2002 EZZ9600I SNTP server ready
| Tue Apr 2 15:28:15 2002 Sending NTP message to multicast address 224.0.1.1
| Tue Apr 2 15:30:15 2002 Sending NTP message to multicast address 224.0.1.1
|

```

Part 4. Appendixes

Appendix A. First Failure Support Technology (FFST)

This appendix contains the following sections:

- FFST probe index
- FFST probe information
- FFST probe naming convention
- FFST probe descriptions

FFST probe index

Table 44 provides an index of FFST probes by probe name and component:

Table 44. FFST probes

Probe name	Component	Reference probes
EZBIEDST	IOCTL Enablement	IOCTL Enablement Probes
EZBPADST	Pascal API	Pascal API Probes
EZBPF DST	PFS IOCTL	PFS IOCTL Probes
EZBTRDST	TELNET Transform	TELNET Transform Probes
EZBT TDST	TELNET SRV	TELNET SRV Probes
EZBCFDST	Configuration Services	Configuration Services Probes
EZBITDST	Infrastructure	Infrastructure
EZBCABND	TCP/IP Base	TCP/IP Base
EZBTC DST	Transmission Control Protocol	Transmission Control Protocol Probes
EZBUDDST	Update Datagram Protocol Layer	Update Datagram Protocol Layer Probes
EZBSKDST	Streams	Streams Probes
EZBRWDST	Raw IP Layer	Raw IP Layer Probes
EZBIPDST	Internet Protocol	Internet Protocol Probes

FFST probe information

When a TCP/IP probe is triggered, an anomaly has occurred in the network. The process that received the condition might not complete normally. The TCP/IP program should attempt to recover from the anomaly and will continue processing subsequent requests. Recovery might not be possible for some system anomalies and subsequent requests might fail, terminals might hang, and other abnormal conditions might occur.

Dump data is collected to assist in finding the source of the problem. Contact the appropriate IBM Support Center and supply the service representative with the console listing that is written at the time the error and the dump data produced by the probe.

FFST probe naming conventions

Table 45 lists the naming conventions for FFST probes used in TCP/IP.

Table 45. FFST naming conventions

Characters	Example	Description
1, 2, 3	EZB	These characters represent the product identifier. For TCP/IP, these characters are EZB.
4, 5	IT	These characters represent the TCP/IP component identifier, IT is the component identifier for Infrastructure Services.
6	C	For TCP/IP, this character is usually a C.
7, 8	01	These characters represent the probe number. This number is not duplicated.

FFST probe descriptions

This section includes a table for each component that contains FFST probe instructions. The components are in alphabetical order, and the probes for each component are in alphanumeric order by probe name. Table 44 on page 703 provides an index of FFST probes in alphanumeric order by probe name. Each table in this section shows the probe name, the module that issued it, and whether the probe creates a full or minidump when triggered.

Table 46 lists the FFST probes for IOCTL enablement (EZBIECxx).

Table 46. IOCTL enablement probes

Probe name	Module	Description	Dump type
EZBIEC01	EZBIEHOM	Logical interface missing	FULL
EZBIEC03	EZBIEPRT	Add Portlist Member Failure	FULL
EZBIEC04	EZBIECTL	IOCTL Command is Not 99	FULL
EZBIEC05	EZBIECTL	Null Queue Pointers	FULL
EZBIEC06	EZBIECTL	Invalid IOCTL Message	FULL
EZBIEC07	EZBIEINI	m_begin Interval Exceeded	FULL

Note: When the EZBIE07 FFST probe is hit, it is recommended that you recycle the TCPIP stack because it is not stable.

Table 47 lists the FFST probes for Infrastructure Services (EZBITCxx).

Table 47. Infrastructure services probes

Probe name	Module	Description	Dump type
EZBITC01	EZBITPCI	Connect entry failure	FULL
EZBITC02	EZBITTUB	Timer cancel for BAD TQE	FULL
EZBITC05	EZBITTUB	Timer cancel for BAD TQE2	FULL
EZBITC07	EZBITDUS	Invalid ASCB	FULL
EZBITC08	EZBITPCT	Entry table destroy failure	FULL
EZBITC09	EZBPTDEF	Pat tree key zero	FULL

Table 47. Infrastructure services probes (continued)

Probe name	Module	Description	Dump type
EZBITC10	EZBPTDEF	Pat tree key too big	FULL
EZBITC11	EZBPTADD	Pat tree key exists	FULL
EZBITC13	EZBITKRA	Lock release error	FULL
EZBITC15	EZBITKRA	Lock release error - DUCB	FULL
EZBITC16	EZBITKRS	Suspend Lock Failure1	FULL
EZBITC17	EZBITKRS	DUCB mismatch	FULL
EZBITC18	EZBITKRS	Lock Suspend Failure2	FULL
EZBITC19	EZBITSCS	Storage size requested error	FULL
EZBITC21	EZBITSMT	Message triple release failure	FULL
EZBITC22	EZBITPCI	Create entry table failure	FULL
EZBITC23	EZBITPCI	TRESERVE linkage index failure	FULL

Table 48 lists the FFST probes for Pascal API (EZBPACxx).

Table 48. FFST probes for Pascal API

Probe name	Module	Description	Dump type
EZBPAC01	EZBPAISL	Streams operation software failure	FULL
EZBPAC02	EZBPAISL	Streams operation software failure	FULL
EZBPAC03	EZBPAMQY	Streams operation software failure	FULL
EZBPAC04	EZBPAMQY	Streams operation software failure	FULL
EZBPAC05	EZBPAPIN	Streams operation software failure	FULL
EZBPAC06	EZBPAPIN	Streams operation software failure	FULL
EZBPAC07	EZBPAPIN	Streams operation software failure	FULL
EZBPAC08	EZBPAPIN	Streams operation software failure	FULL
EZBPAC09	EZBPARCL	Streams operation software failure	FULL
EZBPAC10	EZBPAROP	Streams operation software failure	FULL
EZBPAC11	EZBPAROP	Streams operation software failure	FULL
EZBPAC12	EZBPAROP	Streams operation software failure	FULL
EZBPAC13	EZBPAROP	Streams operation software failure	FULL
EZBPAC14	EZBPAROP	Streams operation software failure	FULL

Table 48. FFST probes for Pascal API (continued)

Probe name	Module	Description	Dump type
EZBPAC15	EZBPAROP	Streams operation software failure	FULL
EZBPAC16	EZBPAROP	Streams operation software failure	FULL
EZBPAC17	EZBPARRV	Streams operation software failure	FULL
EZBPAC18	EZBPARRV	Streams operation software failure	FULL
EZBPAC19	EZBPARRV	Streams operation software failure	FULL
EZBPAC20	EZBPARNN	Streams operation software failure	FULL
EZBPAC21	EZBPART2	Function code error	FULL
EZBPAC22	EZBPATAB	Streams operation software failure	FULL
EZBPAC23	EZBPATAB	Streams operation software failure	FULL
EZBPAC24	EZBPATAB	Streams operation software failure	FULL
EZBPAC25	EZBPASTR	Invalid type of M_ERROR	FULL
EZBPAC26	EZBPASTR	Storage allocation failure	FULL
EZBPAC27	EZBPASTR	Unsupported option	FULL
EZBPAC28	EZBPASTR	Unsupported option	FULL
EZBPAC29	EZBPASTR	Unrecognized TPI	FULL
EZBPAC30	EZBPASTR	Streams operation software failure	FULL
EZBPAC31	EZBPASTR	Streams operation software failure	FULL
EZBPAC32	EZBPASTR	Storage allocation failure	FULL
EZBPAC33	EZBPASTR	Streams operation software failure	FULL
EZBPAC34	EZBPASTR	Streams operation software failure	FULL
EZBPAC35	EZBPASTR	Streams operation software failure	FULL
EZBPAC36	EZBPASTR	Streams operation software failure	FULL
EZBPAC37	EZBPASTR	Streams operation software failure	FULL
EZBPAC38	EZBPASTR	Streams operation software failure	FULL
EZBPAC39	EZBPASTR	Streams operation software failure	FULL
EZBPAC40	EZBPASTR	Streams operation software failure	FULL

Table 48. FFST probes for Pascal API (continued)

Probe name	Module	Description	Dump type
EZBPAC41	EZBPASTR	Streams operation software failure	FULL
EZBPAC42	EZBPASTR	Streams operation software failure	FULL
EZBPAC43	EZBPASTR	Storage allocation failure	FULL
EZBPAC44	EZBPASTR	Storage allocation failure	FULL
EZBPAC45	EZBPASTR	Storage allocation failure	FULL
EZBPAC46	EZBPAUCL	Streams operation software failure	FULL
EZBPAC47	EZBPAUNR	Streams operation software failure	FULL
EZBPAC48	EZBPAUNR	Streams operation software failure	FULL
EZBPAC49	EZBPAUNR	Streams operation software failure	FULL
EZBPAC50	EZBPAURV	Streams operation software failure	FULL
EZBPAC51	EZBPAURV	Streams operation software failure	FULL
EZBPAC52	EZBPAURV	Streams operation software failure	FULL
EZBPAC53	EZBPATOP	Streams operation software failure	FULL
EZBPAC54	EZBPATOP	Streams operation software failure	FULL
EZBPAC55	EZBPATOP	Streams operation software failure	FULL
EZBPAC56	EZBPATOP	Streams operation software failure	FULL
EZBPAC57	EZBPATOP	Streams operation software failure	FULL
EZBPAC58	EZBPATOP	Streams operation software failure	FULL
EZBPAC59	EZBPATOP	Streams operation software failure	FULL
EZBPAC60	EZBPAUOP	Streams operation software failure	FULL
EZBPAC61	EZBPAUOP	Streams operation software failure	FULL
EZBPAC62	EZBPAUOP	Streams operation software failure	FULL
EZBPAC63	EZBPAUOP	Streams operation software failure	FULL
EZBPAC64	EZBPAUOP	Streams operation software failure	FULL
EZBPAC65	EZBPAUOP	Streams operation software failure	FULL

Table 48. FFST probes for Pascal API (continued)

Probe name	Module	Description	Dump type
EZBPAC66	EZBPAUOP	TPI protocol error	FULL
EZBPAC67	EZBPATFR	Streams operation software failure	FULL
EZBPAC68	EZBPATFR	Streams operation software failure	FULL
EZBPAC69	EZBPATOA	Streams operation software failure	FULL
EZBPAC70	EZBPATOA	Streams operation software failure	FULL
EZBPAC71	EZBPATOA	Streams operation software failure	FULL
EZBPAC72	EZBPATOA	Streams operation software failure	FULL
EZBPAC73	EZBPATSN	Streams operation software failure	FULL
EZBPAC74	EZBPATST	Streams Operation Software Error	FULL
EZBPAC75	EZBPATTN	Streams operation software failure	FULL
EZBPAC76	EZBPATTN	Streams operation software failure	FULL
EZBPAC77	EZBPATTN	Streams operation software failure	FULL
EZBPAC78	EZBPAUSN	Streams operation software failure	FULL
EZBPAC79	EZBPAUST	Streams operation software failure	FULL
EZBPAC80	EZBPAUST	Streams operation software failure	FULL
EZBPAC81	EZBPATCL	Streams operation software failure	FULL
EZBPAC82	EZBPATCL	Allocate storage failure	FULL
EZBPAC83	EZBPATCL	Streams operation software failure	FULL
EZBPAC84	EZBPATCL	Allocate storage failure	FULL
EZBPAC85	EZBPATON	Streams Software Operation Error	FULL
EZBPAC86	EZBPATON	Streams operation software failure	FULL
EZBPAC87	EZBPATON	Streams operation software failure	FULL
EZBPAC88	EZBPATON	Streams operation software failure	FULL
EZBPAC89	EZBPATON	Streams operation software failure	FULL
EZBPAC90	EZBPATON	Streams operation software failure	FULL

Table 48. FFST probes for Pascal API (continued)

Probe name	Module	Description	Dump type
EZBPAC91	EZBPATON	Streams operation software failure	FULL
EZBPAC92	EZBPATON	Streams operation software failure	FULL
EZBPAC93	EZBPATON	Streams operation software failure	FULL
EZBPAC94	EZBPATON	Streams operation software failure	FULL
EZBPAC95	EZBPATON	TPI protocol error	FULL
EZBPAC96	EZBPATON	Streams operation software failure	FULL
EZBPAC97	EZBPATON	Streams operation software failure	FULL
EZBPAC98	EZBPATON	TPI protocol error	FULL
EZBPAC99	EZBPATON	Streams operation software failure	FULL
EZBPAC0A	EZBPATON	Streams operation software failure	FULL
EZBPAC0B	EZBPATON	TPI protocol error	FULL
EZBPAC0C	EZBPATON	Streams operation software failure	FULL
EZBPAC0D	EZBPATON	Streams operation software failure	FULL
EZBPAC0E	EZBPATON	TPI protocol error	FULL
EZBPACA0	EZBPATOP	Streams operation software failure	FULL
EZBPACA1	EZBPATOP	Streams operation software failure	FULL
EZBPACA2	EZBPATOP	Streams operation software failure	FULL
EZBPACB0	EZBPASTR	Storage allocate failure	FULL

Table 49 lists the FFST probes for PFS IOCTL (EZBPFCxx).

Table 49. PFS IOCTL probes

Probe name	Module	Description	Dump type
EZBPFC01	EZBPFIOC	SIOCSETTKN mismatch	FULL
EZBPFC02	EZBPFIOC	SIOCSETTKN mismatch	FULL

Table 50 lists the FFST probes for Telnet Transform (EZBTRCxx).

Table 50. Telnet transform probes

Probe name	Module	Description	Dump type
EZBTRC01	EZBTRCLT	Unexpected transform request	FULL
EZBTRC03	EZBTRGTI	Terminal ID mismatch	FULL
EZBTRC04	EZBTRMST	Unexpected transform WorkQ request	FULL
EZBTRC05	EZBTRRTI	Negative transform terminal value	FULL

Table 51 lists the FFST probes for Telnet SRV (EZBTTCxx).

Table 51. FFST probes for Telnet SRV

Probe name	Module	Description	Dump type
EZBTTC01	EZBTTCCLS	Unlocatable server/vector table	FULL
EZBTTC02	EZBTTCCLS	CVB lock failure	FULL
EZBTTC03	EZBTTCFLT	Invalid TCVB token range	FULL
EZBTTC04	EZBTTCFLT	Invalid TST entry	FULL
EZBTTC05	EZBTTCFLT	Telnet token segment table not found	FULL

Table 52 lists FFST probes for Configuration Services (EZBCFCxx).

Table 52. Configuration services probes

Probe name	Module	Description	Dump type
EZBCFC01	EZACFFST	Unknown configuration error	FULL
EZBCFC02	EZACFTEL	Bad protocol Type 1	FULL
EZBCFC03	EZACFFST	Configuration bad parameters error	FULL
EZBCFC04	EZACFTEL	Socket closed	FULL
EZBCFC05	EZACFTEL	Bad protocol Type 2	FULL
EZBCFC06	EZACFTEL	Bad protocol Type 3	FULL

Table 53 lists the FFST probe for TCP/IP Base (EZBABCxx).

Table 53. TCP/IP Base probes

Probe name	Module	Description	Dump type
EZBABC01	EZBCABND	A C abend recovery failed	FULL

Table 54 on page 711 lists the FFST probes for Transmission Control Protocol (EZBTCCxx).

Table 54. Transmission Control Protocol probes

Probe name	Module	Description	Dump type
EZBTCC01	EZBTCSTR	Name on Open Does Not Match	FULL
EZBTCC02	EZBTCSTR	Could not allocate the SID	FULL
EZBTCC03	EZBTCSTR	Cannot Repeat Named Open	FULL
EZBTCC04	EZBTCSTR	Hashtable Insert Failure	FULL
EZBTCC05	EZBTCWRT	Not the Controlling Stream	FULL
EZBTCC06	EZBTCWRT	Not the Controlling Stream	FULL
EZBTCC07	EZBTCWRT	Not the Controlling Stream	FULL

Table 55 lists the FFST probes for Update Datagram Protocol Layer (EZBUDCxx).

Table 55. Update Datagram Protocol Layer probes

Probe name	Module	Description	Dump type
EZBUDC01	EZBUDEXC	DMUX Machine Index Failure	FULL
EZBUDC02	EZBUDEXC	DMUX Machine Index Failure	FULL
EZBUDC03	EZBUDEXC	SNMP Machine Index Failure	FULL
EZBUDC04	EZBUDSTR	Name on Open Does Not Match	FULL
EZBUDC05	EZBUDSTR	Allocate the MUCB SID failure	FULL
EZBUDC06	EZBUDSTR	Stack is Already Active	FULL
EZBUDC07	EZBUDSTR	Unlock for Machine Index Failure	FULL
EZBUDC08	EZBUDSTR	Unlock for Machine Index Failure	FULL
EZBUDC09	EZBUDSTR	Unlock for Machine Index Failure	FULL
EZBUDC10	EZBUDWRT	Unknown Primitive Error Exit	FULL
EZBUDC11	EZBUDWRT	Unknown Primitive Error Exit	FULL
EZBUDC12	EZBUDWRE	Matching Prefix Error	FULL
EZBUDC13	EZBUDWRE	Matching Prefix Error	FULL

Table 56 lists the FFST probes for Streams (EZBSKCxx).

Table 56. Streams probes

Probe name	Module	Description	Dump type
EZBSKC01	EZBSKVRB	Streams Are Not Functioning (TSDX_Streams_vcastint)	FULL
EZBSKC02	EZBSKVRB	Unsupported Message Type	FULL

Table 57 lists the FFST probes for Raw IP Layer (EZBRWCxx).

Table 57. Raw IP Layer probes

Probe name	Module	Description	Dump type
EZBRWC01	EZBRWWRI	WILD TPI Primitive to RAW	FULL
EZBRWC02	EZBRWWRI	Invalid Messages	FULL
EZBRWC03	EZBRWSTR	Name on Open Does Not Match	FULL
EZBRWC04	EZBRWSTR	Could Not Allocate the MRCB SID	FULL
EZBRWC05	EZBRWSTR	Stack is Already Active	FULL

Table 58 lists the FFST probes for Internet Protocol (EZBIPCxx).

Table 58. FFST probes for Internet Protocol

Probe name	Module	Description	Dump type
EZBIPC01	EZBIPSTR	Not a Clone Open	FULL

Table 59 lists the FFST probes for the Cross-System Coupling Facility (XCF) (EZBXFCxx).

Table 59. XCF probes

Probe name	Module	Description	Dump type
EZBXFC01	EZBXFINI	Join Failed	FULL
EZBXFC02	EZBXFINI	Second Query Failed	FULL
EZBXFC03	EZBXFINI	First Query Failed	FULL
EZBXFC04	EZBXFMSI	MsgI Failed	FULL
EZBXFC05	EZBXFMSO	MsgO Failed	FULL

Note: When partitioning systems out of the sysplex, FFST problem EZBXFC05 may be seen on active systems in the sysplex. This can occur when a response is not given to IXC402D in a timely manner. To avoid this, it is suggested you setup the SFM policy to automatically partition systems from the sysplex without having to respond to IXC402D. Refer to *z/OS MVS Setting Up a Sysplex* for information on setting up the SFM policy.

Appendix B. Overview of internetworking

Networking with TCP/IP connects different networks so that they form one logical interconnected network. This large overall network is called an *internetwork*, or more commonly, an *intranet* or *internet*. Each network uses its own physical layer, and the different networks are connected to each other by means of machines that are called *gateways*.

Gateways transfer IP datagrams between networks. This function is called *routing*; therefore, the internet gateways are often called *routers*. Within this appendix, the terms router and gateway are synonymous; both refer to a machine that transfers IP datagrams between different networks.

Note: If IP datagrams are not passed properly over a bridge, none of the higher TCP/IP protocols or applications will work correctly. For a discussion of bridges, refer to *TCP/IP Tutorial and Technical Overview*.

Linking networks in this way takes place at the network level of the International Organization for Standardization (ISO). It is possible to link networks at a lower level layer using *bridges*. Bridges link networks at the ISO data link layer. Bridges pass packets or frames between different physical networks regardless of the protocols contained within them. An example of a bridge is the IBM 8209, which can interconnect an Ethernet network and a token-ring network.

Note: A bridge does *not* connect TCP/IP networks together. It connects physical networks together that will still form the same TCP/IP network. (A bridge does *not* do IP routing.)

Figure 74 on page 714 depicts a router and a bridge. The router connects Network 1 to Network 2 to form an intranet.

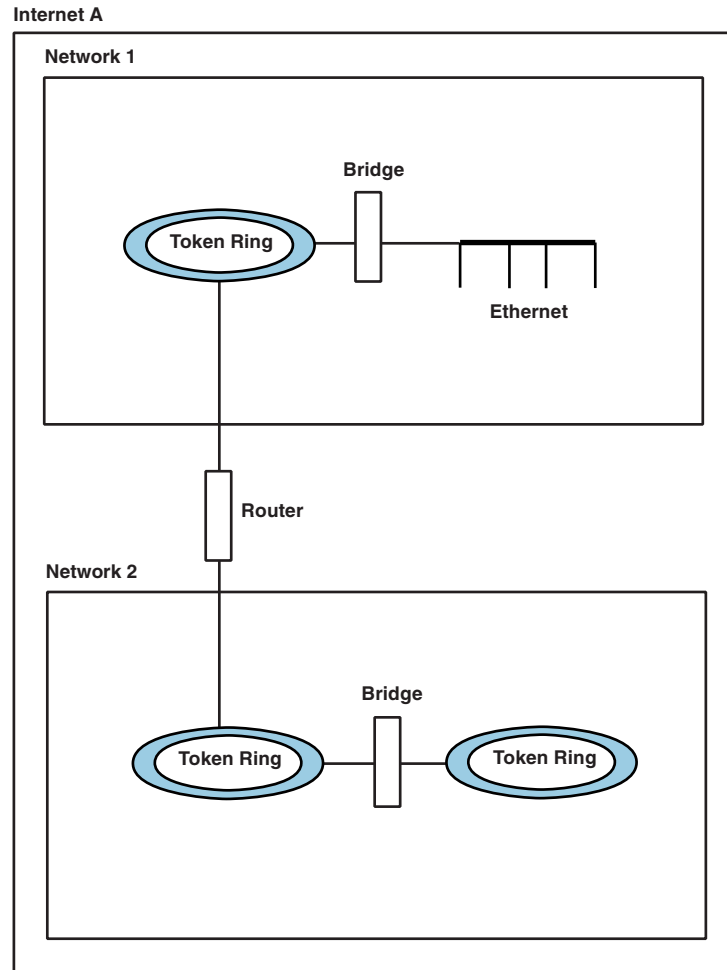


Figure 74. Routers and bridges within an internet

Maximum transmission unit (MTU)

Different physical networks have different maximum frame sizes. Within the different frames, there is a maximum size for the data field. This value is called the *maximum transmission unit* (MTU), or maximum packet size in TCP/IP terms.

Figure 75 on page 715 shows the relationship between MTU and frame size.

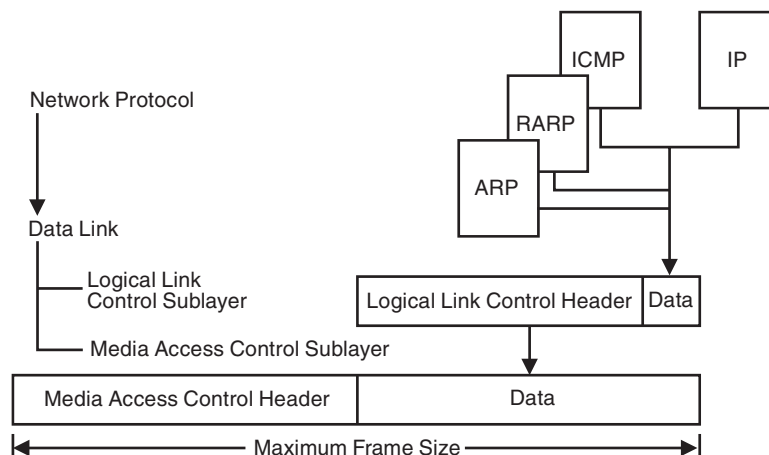


Figure 75. Relationship of MTU to frame size

If an IP datagram is to be sent out onto the network and the size of the datagram is bigger than the MTU, IP will fragment the datagram into multiple fragments, so that it will fit within the data fields of the frames. If the MTU is larger than the network can support, then the data is lost.

The value of MTU is especially important when bridging is used because of the different network limits. *RFC 791 —Internet Protocols* states that all IP hosts must be prepared to accept datagrams of up to 576 bytes.

You can configure an MTU using the `max_packet_size` value on the `GATEWAY` statement or the `MTU` parameter on the `BEGINROUTES` statement.

Fiber Distributed Data Interface (FDDI)

The FDDI specifications define a family of standards for 100 Mbps fiber optic LANs that provide the physical layers and media access control sublayer of the data link layer, as defined by the ISO/OSI Model.

IP-FDDI defines the encapsulating of IP datagrams and ARP requests and replies in FDDI frames.

All frames are transmitted in standard IEEE 802.2 LLC Type 1 Unnumbered Information format, with the DSAP and SSAP fields of the 802.2 header set to the assigned global SAP value for SNAP (decimal 170). The 24-bit Organization Code in the SNAP header is set to zero, and the remaining 16 bits are the EtherType from Assigned Numbers:

- 2048 for IP
- 2054 for ARP

Typically, the MTU is set to 4352.

Mapping of 32-bit internet addresses to 48-bit FDDI addresses is done by the ARP dynamic discovery procedure. The broadcast internet addresses (whose <host address> is set to all ones) are mapped to the broadcast FDDI addresses (all ones).

IP datagrams are transmitted as a series of 8-bit bytes using the usual TCP/IP transmission order called “big-endian” or “network byte order.”

For more information on FDDI architecture, refer to *LAN Concepts and Products*.

Token-Ring IEEE 802.5

When a token-ring frame passes through a bridge, the bridge adds information to the routing information field (RIF) of the frame (assuming that the bridge supports source route bridging). The RIF contains information concerning the route taken by the frame and, more importantly, the maximum amount of data that the frame can contain within its data field. This is called the maximum information field (I-field). The value specified for the maximum I-field is sometimes referred to as the largest frame size, but this means the largest frame size, *excluding* headers. See Figure 76 for details on the relationship of the I-field to the header fields.

Note: It is important to be aware that the IBM implementation limits the number of bridges through which a frame can be passed to seven. An attempt to pass a frame through an eighth bridge will fail.

The maximum I-field is always decreased by a bridge when it cannot handle the value specified. So, for a given path through a number of token-ring bridges, the maximum I-field is the largest value that *all* of the bridges will support. This value is specified in the Routing Control (RC) field within the RIF as shown in Figure 76.

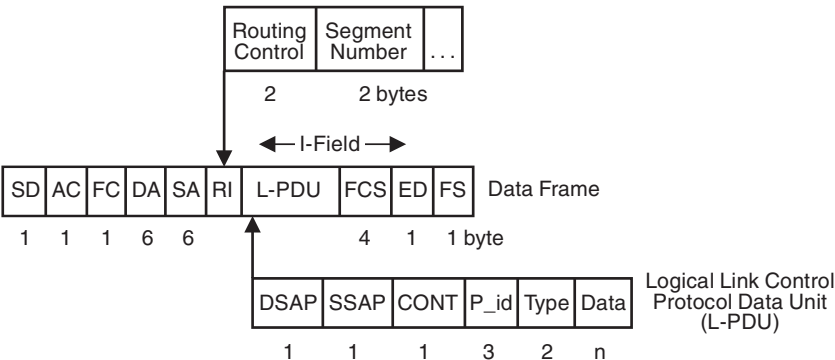


Figure 76. Format of an IEEE 802.5 token-ring frame

The size of the MTU is the maximum amount of data that is allowed within a frame. The token-ring architecture specifies the maximum value of the I-field in the data frame, which corresponds to the maximum size of the L-PDU. The maximum I-field value is determined by the bit configuration in the RC field, and is present in all routed frames.

Table 60 on page 717 shows the relationship between the RC field and the maximum I-field values.

Table 60. Relationship between RC field and maximum I-field value

Routing control field	Maximum I-field in bytes
x000 xxxx xxxx xxxx	516
x001 xxxx xxxx xxxx	1500
x010 xxxx xxxx xxxx	2052
x011 xxxx xxxx xxxx	4472
x100 xxxx xxxx xxxx	8144
x101 xxxx xxxx xxxx	11407
x110 xxxx xxxx xxxx	17800

Figure 76 on page 716 shows that, within the L-PDU, the Logical Link Control (LLC) header uses eight bytes. Thus the MTU value is eight bytes less than the maximum I-field. Note that the L-PDU contains a SNAP header, as described in “Subnetwork Access Protocol (SNAP)” on page 718. Follow this example to calculate the MTU for a token-ring. The token-ring bridges always adjust the value of the maximum I-field to that of the smallest one in the path. Ensure that the MTU value is less than the value specified by the bridge.

Typically, within a 4-Mbps token-ring network, the value of maximum I-field will be 2052 bytes. Therefore, the MTU would be set to 2044 bytes (2052 minus eight bytes for the LLC header).

IEEE 802.3

The frame used in IEEE 802.3 Ethernet networks is shown in Figure 77.

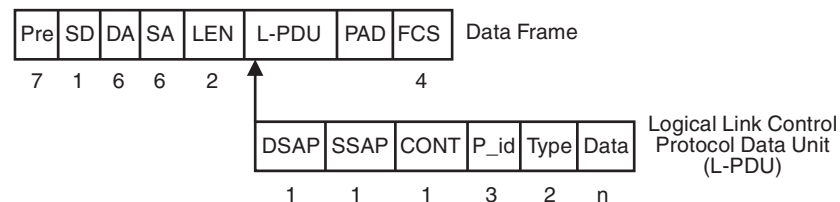


Figure 77. Format of an IEEE 802.3 frame

The maximum size of the L-PDU for a 10Mbps network is 1500 bytes. Because eight bytes are used within the L-PDU for the LLC header, this means that the maximum size of the data field is 1492 bytes. Therefore, the MTU for IEEE 802.3 networks should be set to 1492 bytes.

Ethernet — DIX V2

The frame used in DIX Ethernet networks is shown in Figure 78 on page 718.

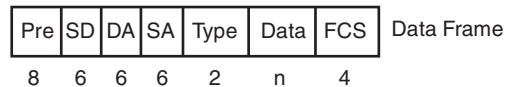


Figure 78. Format of an Ethernet V2 frame

There is no LLC data in an Ethernet V2 frame. The maximum size for the frame is 1526 bytes. This means that the data field can be 1500 bytes maximum. The MTU for Ethernet V2 can be set to 1500 bytes.

It is possible to bridge Ethernet V2 frames to either IEEE 802.3 or IEEE 802.5 networks; an LLC header is added or removed from the frame, as required, as part of the conversion when bridging.

Subnetwork Access Protocol (SNAP)

The TCP/IP software provides protocol support down to the ISO network layer. Following this layer is the data link layer, which can be separated into two sublayers. These are the *Logical Link Control* (LLC) and the *Media Access Control* (MAC) layers.

The IEEE 802.2 standard defines the LLC sublayer, and the MAC sublayer is defined in IEEE 802.3, IEEE 802.4, and IEEE 802.5.

The format of an IEEE 802.2 LLC header with the SNAP header is shown in Figure 79.

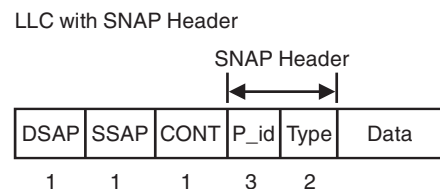


Figure 79. SNAP header

The values of the fields in the LLC header when a SNAP header is used are specified in *RFC 1042 - Standard for Transmission of IP Datagrams over IEEE 802 Networks*. The values specified are:

Field **Value**

DSAP X'AA'

SSAP X'AA'

CONT X'03' Specifies unnumbered information (UI)

P_id X'00 00 00'

Type

X'08 00' — IP

X'08 06' — ARP

X'08 35' — RARP

IP routing

IP routing is based on routing tables held within a router or internet host. These tables contain routes which can either be *static* or *dynamic*. Typically, static routes are predefined within a configuration file, and dynamic routes are “learned” from the network, using a *routing* protocol.

Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6)

There are two Internet protocols used to assign addresses to links on a host, Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6). The majority of current internets use IPv4. This protocol is nearly 20 years old and is approaching the limits of the node addresses that its 32 bit addresses allow. IPv6 is the next generation of the Internet Protocol, designed to replace IPv4. Among other advantages, the 128 bit addresses defined by IPv6 provide nearly limitless addresses.

Although IPv6 is expected to eventually replace IPv4, it is likely that they will coexist for a number of years during the transition.

Internet Protocol Version 4 (IPv4)

A link on a host on an intranet is identified by its *IP address*. *Internet Protocol* (IP) is the protocol that is used to deliver datagrams between such hosts. It is assumed the reader is familiar with the TCP/IP protocols. Details of some of the protocols can be found in the *TCP/IP Tutorial and Technical Overview*. Specific information relating to the Internet Protocol can be found in RFC 791.

An IPv4 address is a 32-bit address that is usually represented in dotted decimal notation, with a decimal value representing each of the four octets (bytes) that make up the address. For example:

```
|          00001001010000110110000100000010      32-bit address
|          00001001 01000011 01100001 00000010      4 octets
|              9      67      97      2      dotted decimal notation (9.67.97.2)
```

The IPv4 address consists of a *network address* and a *host address*. Within the Internet, the network addresses are assigned by a central authority, the *Network Information Center* (NIC). The portion of the IPv4 address that is used for each of these addresses is determined by the class of address. There are three commonly used classes of IPv4 addresses (see Figure 80).

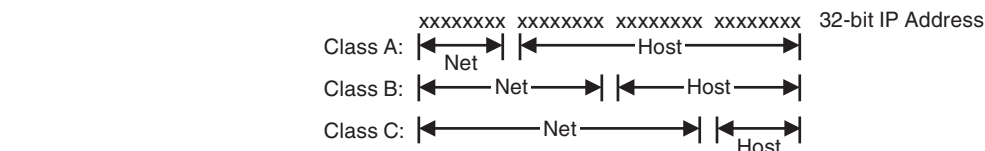


Figure 80. Classes of IPv4 addresses

The class of the address is determined by the first octet of the IPv4 address. Figure 81 on page 720 shows how the class of address is determined. The figure also shows Class D addresses. Class D addresses represent multicast groups, not

network IP addresses. Multicast group addresses consist of the high-order, four bits of 1110 and the remaining 28 bits, which form a multicast group ID.

32-bit address		xxxxxxx xxxxxxx xxxxxxx xxxxxxx
Class A		0xxxxxxx xxxxxxx xxxxxxx xxxxxxx
	min	00000000
	max	01111111
	range	1 - 126 (decimal notation; 0 and 127 are reserved)
Class B		10xxxxxx xxxxxxx xxxxxxx xxxxxxx
	min	10000000
	max	10111111
	range	128 - 191 (decimal notation)
Class C		110xxxxx xxxxxxx xxxxxxx xxxxxxx
	min	11000000
	max	11011111
	range	192 - 223 (decimal notation)
Class D		1110xxxx xxxxxxx xxxxxxx xxxxxxx
	min	11100000
	max	11101111
	range	224-239 (decimal notation)

Figure 81. Determining the class of an IPv4 address

As shown in Figure 81, the value of the bits in the first octet determine the class of address, and the class of address determines the range of values for the network and host segment of the IPv4 address. For example, the IPv4 address 9.67.97.2 would be a class A address, since the first two bits in the first octet contain B'00'. The network part of the IPv4 address is "9" and the host part of the IPv4 address is "67.97.2".

Refer to *RFC 1166 — Internet Numbers* for more information about IPv4 addresses. Refer to *RFC 1060 — Assigned Numbers* for more information about reserved network and host IPv4 addresses, such as a *network broadcast address*.

Internet Protocol Version 6 (IPv6)

As described above, IPv4 addresses are represented in dotted-decimal format. The 32-bit address is divided along 8-bit boundaries. Each set of 8 bits is converted to its decimal equivalent and separated by periods. In contrast, IPv6 addresses are 128-bits divided along 16-bit boundaries. Each 16-bit block is converted to a 4-digit hexadecimal number and separated by colons. The resulting representation is called colon-hexadecimal.

There are three conventional forms for representing IPv6 addresses as text strings:

The preferred form is x:x:x:x:x:x:x, where x is the hexadecimal value of the eight 16-bit pieces of the address. For example:

```
FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
1080:0:0:0:8:800:200C:417A
```

Note:

It is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field. The following is the only exception.

It is common in some styles of IPv6 addresses to contain long strings of zero bits. To make writing addresses containing zero bits easier, a special syntax is available to compress the zeros. Use two colons (::) to indicate multiple groups of 16 bits of zeros. The two colons (::) can appear only once in an address. The two colons (::) can also be used to compress the leading zeros, the trailing zeros, or both in an address.

For example, the following addresses:

1080:0:0:0:8:800:200C:417A	a unicast address
FF01:0:0:0:0:0:0:101	a multicast address
0:0:0:0:0:0:0:1	the loopback address
0:0:0:0:0:0:0:0	the unspecified addresses

can be represented as:

1080::8:800:200C:417A	a unicast address
FF01::101	a multicast address
::1	the loopback address
::	the unspecified addresses

An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:d.d.d.d, where x is the hexadecimal value of the six high-order 16-bit pieces of the address, and d is the decimal value of the four low-order 8-bit pieces of the address (standard IPv4 representation). For example, 0:0:0:0:0:0:13.1.68.3 can be expressed in condensed form as ::13.1.68.3

Figure 82 on page 722 shows a simple network with a bridge and a router.

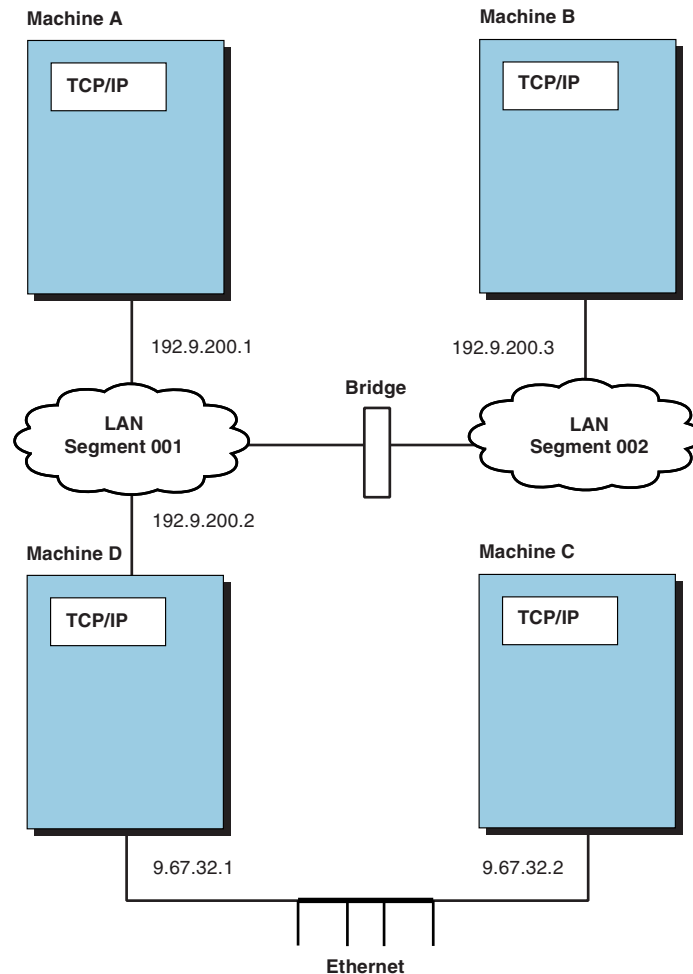


Figure 82. Routing and bridging

Machine D is acting as an IP router and will transfer IP datagrams between the class C, 192.9.200, network and the class A, 9.67.32 network. It is important to note that for Machine B to communicate with Machine C using TCP/IP, both Machine D and the bridge have to be correctly configured and working.

TCP/IP uses the HOME statements, defined in the data set *hlq.PROFILE.TCPIP*, to assign home addresses and associated link names. HOME statements can be updated using the VARY TCPIP command. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about both the HOME statements and the VARY TCPIP command.

Direct routing

Direct routing can take place when two hosts are directly connected to the same physical network. This can be a bridged token-ring network, a bridged Ethernet, or a bridged token-ring network and Ethernet. The distinction between direct routing and indirect routing is that, with direct routing, an IP datagram can be delivered to the remote host without subsequent interpretation of the IP address, by an intermediate host or router.

In Figure 82 on page 722, a datagram traveling from Machine A to Machine B would be using direct routing, although it would be traveling through a bridge.

Indirect routing

Indirect routing takes place when the destination is *not* on a directly attached IP network, forcing the sender to forward the datagram to a router for delivery.

In Figure 82 on page 722, a datagram from Machine A being delivered to Machine C would be using indirect routing, with Machine D acting as the router (or gateway).

Simplified IP datagram routing algorithm

To route an IP datagram on the network, the algorithm shown in Figure 83 is used.

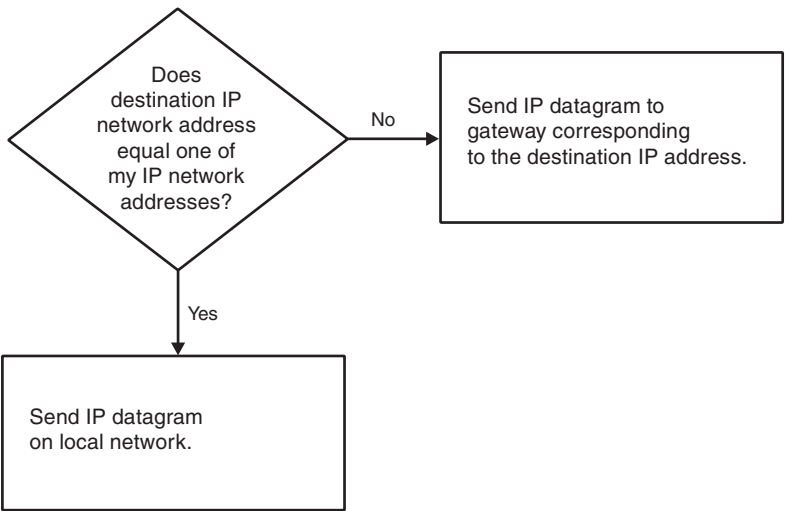


Figure 83. General IP routing algorithm

Using this general routing algorithm, it is very easy to determine where an IP datagram will be routed. Following is a simple example based on the configuration shown in Figure 82 on page 722.

Machine A IP Address = 192.9.200.1

Routing Table

Destination	Gateway	
192.9.200.1	192.9.200.1	(Machine A's network interface)
9.0.0.0	192.9.200.2	(Route to the 9.n.n.n address is via Machine D, 192.9.200.2)

Machine A sends a datagram to host 192.9.200.3 (Machine B), using the direct route, 192.9.200.1 (its own network interface). Machine A sends a datagram to host 9.67.32.2 (Machine C), using the indirect route, 192.9.200.2 (Machine D), and Machine D then forwards the datagram to Machine C.

IPv4 Subnetting

IPv4 allows for a variation of the network and host segments of an IP address, known as *subnetting*, can be used to physically and logically design a network. For example, an organization can have a single internet network address (NETID) that is known to users outside the organization, yet configure its internal network into different departmental subnets. Subnetwork addresses enhance local routing capabilities, while reducing the number of network addresses required.

To illustrate this, consider a simple example. Assume that we have an assigned class C network address of 192.9.200 for our site. This would mean that we could have host addresses from 192.9.200.1 to 192.9.200.254. If we did not use subnetting, then we could only implement a single IP network with 254 hosts. To split our site into two logical subnetworks, we could implement the network scheme shown in Figure 84:

Without Subnetting:

	Network Address	Host Address Range
192 9 200 host 11000000 00001001 11001000 xxxxxxxx	192.9.200	1 - 254

With Subnetting:

	Subnet Address	Host Address Range	Subnet Value
192 9 200 64 host 11000000 00001001 11001000 01xxxxxx	192.9.200.64	65 - 126	01
192 9 200 128 host 11000000 00001001 11001000 10xxxxxx	192.9.200.128	129 - 190	10

The subnet mask would be

255	255	255	192
11111111	11111111	11111111	11000000

Figure 84. Subnetting scheme

OS/390 TCP/IP uses a slightly different scheme for the *subnet mask* when defining the GATEWAY statements in the *hlq.PROFILE.TCPIP* data set and for displaying the subnet mask within a *onetstat -g* command. The subnet mask is applied only to the host segment of the IP address, and *onetstat* displays the subnet mask for only the host segment of the IP address. The subnet mask in the preceding chart as defined for OS/390 TCP/IP would be:

0	0	0	192	0.0.0.192
00000000	00000000	00000000	11000000	

Although OS/390 TCP/IP defines the subnet mask differently, the application of the subnet mask and subnet value to the IP address is consistent with RFC-architected routing algorithms. A subnet mask of 255 is used for the remainder of this section of the chapter, to retain symmetry with other routing documents that use 255 as the subnet value for the network segment of an IP address.

Because subnets B'00' and B'11' are both reserved, only two subnets are available. All 0s and all 1s have a special significance in internet addressing and should be used with care. Also notice that the total number of host addresses that we can use is reduced for the same reason. For instance, we cannot have a host address of 16 because this would mean that the subnet/host segment of the address would be B'0001000', which with the subnet mask we are using, would mean a subnet value of B'00', which is reserved.

The same is true for the host segment of the fourth octet. A fourth octet value of B'01111111' is reserved because, although the subnet of B'01' is valid, the host value of B'1' is reserved.

The network segment of the subnet mask is always assumed to be one, so each octet has a decimal value of 255. For example, with a class B address, the first two octets are assumed to be 255.255.

IPv6 Prefixes

The IPv6 prefix concept is similar to IPv4 subnetting. An IPv6 address with a prefix is written as an IPv6 address followed by a decimal number representing the number of bits in the address that constitute the prefix. It is written as:

ipv6-address/prefix-length

where:

ipv6-address

is an IPv6 address in any notation

prefix-length

is a decimal value specifying how many of the leftmost contiguous bits of the address comprise the prefix.

For example, the following are legal representations of the 60-bit prefix 12AB00000000CD3 (hexadecimal):

12AB:0000:0000:CD30:0000:0000:0000:0000/60
12AB::CD30:0:0:0:0/60
12AB:0:0:CD30::/60

When writing both a node address and a prefix of that node address (for example, the node subnet prefix), the two can be combined as follows:

The node address

12AB:0:0:CD30:123:4567:89AB:CDEF

and its subnet number

12AB:0:0:CD30::/60

can be abbreviated as

12AB:0:0:CD30:123:4567:89AB:CDEF/60

Simplified IP datagram routing algorithm with subnets

When subnetting is used, the algorithm required to find a route for an IP datagram is similar to the one for general routing, with the exception that the addresses being compared are the result of a logical AND of the subnet mask and the IP address.

For example:

IP address:	9.67.32.18	00001001	01000011	00100000	00010010
				<AND>	
Subnet Mask:	255.255.255.240	11111111	11111111	11111111	11110000
Result of					
Logical AND:	9.67.32.16	00001001	01000011	00100000	00010000

The subnet address is 9.67.32.16, and it is this value that is used to determine the route used.

Figure 85 shows the routing algorithm used with subnets.

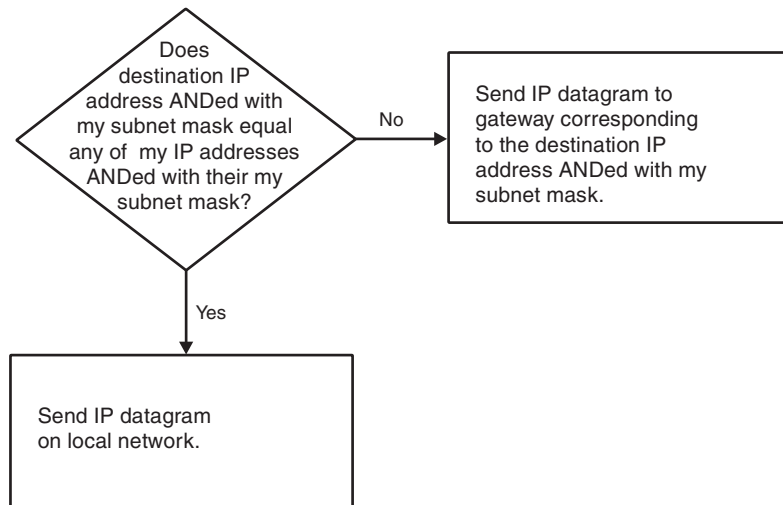


Figure 85. Routing algorithm with subnets

Figure 86 on page 727 shows how a subnet route is resolved.

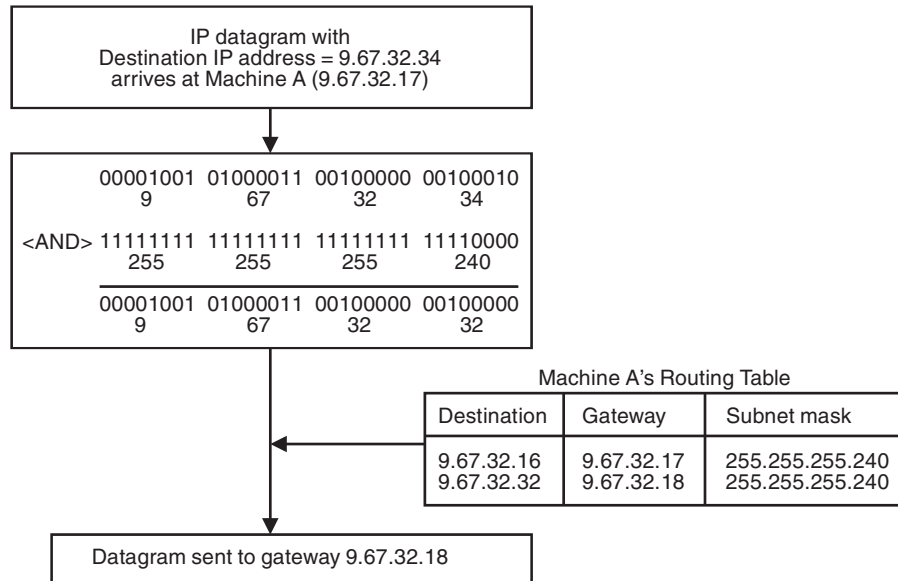


Figure 86. Example of resolving a subnet route

Static routing

Static routing, as the name implies, is defined within the local host, and must be manually changed as the network changes. Typically, a configuration file will contain the definitions for directly-attached networks, routes for specific hosts, and a possible default route that directs packets to a destination for networks that are not previously defined.

Static routes can be defined using either the z/OS TCP/IP GATEWAY or BEGINROUTES statements to configure the internal routing tables; these statements are defined in the *hlq.PROFILE.TCPIP* data set. The internal routing tables for z/OS TCP/IP can be modified by either: 1) changing the GATEWAY or BEGINROUTES statements and recycling the TCP/IP address space or 2) using the VARY TCPIP command. Refer to the *z/OS Communications Server: IP Configuration Reference* for details about defining the GATEWAY or BEGINROUTES statements and using the VARY command.

Note: When the GATEWAY or BEGINROUTES statements are updated using VARY TCPIP, all previously defined static routes are discarded and replaced by the new GATEWAY or BEGINROUTES definitions.

Dynamic routing

Dynamic routing is the opposite of static routing. A TCP/IP protocol is used to dynamically update the internal routing tables when changes to the network occur.

IPv4

For IPv4, there are two dynamic routing protocols available. One routing protocol is the Routing Information Protocol (RIP). It is implemented by both the OROUTED and OMROUTE routing applications. A newer protocol is open shortest path first

(OSPF). It is implemented by OMPROUTE only. For more details about OMPROUTE, see Chapter 25, “Diagnosing OMPROUTE problems” on page 601. For more details about OROUTED, see Chapter 24, “Diagnosing OROUTED problems” on page 581. For configuration information about both applications, refer to the *z/OS Communications Server: IP Configuration Reference*.

IPv6

For IPv6, dynamic routing is performed by the Router Discovery protocol. For more information about IPv6 dynamic routing, refer to the *z/OS Communications Server: IP Configuration Reference*.

Appendix C. How to read a syntax diagram

This syntax information applies to all commands and statements included in this document that do not have their own syntax described elsewhere in this document.

The syntax diagram shows you how to specify a command so that the operating system can correctly interpret what you type. Read the syntax diagram from left to right and from top to bottom, following the horizontal line (the main path).

Symbols and punctuation

The following symbols are used in syntax diagrams:

Symbol	Description
➤	Marks the beginning of the command syntax.
▶	Indicates that the command syntax is continued.
	Marks the beginning and end of a fragment or part of the command syntax.
⏏	Marks the end of the command syntax.

You must include all punctuation such as colons, semicolons, commas, quotation marks, and minus signs that are shown in the syntax diagram.

Parameters

The following types of parameters are used in syntax diagrams.

Required

Required parameters are displayed on the main path.

Optional

Optional parameters are displayed below the main path.

Default

Default parameters are displayed above the main path.

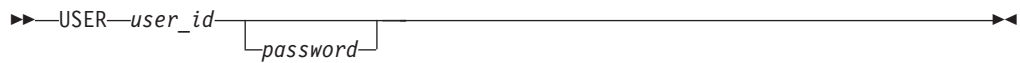
Parameters are classified as keywords or variables. For the TSO and MVS console commands, the keywords are not case sensitive. You can code them in uppercase or lowercase. If the keyword appears in the syntax diagram in both uppercase and lowercase, the uppercase portion is the abbreviation for the keyword (for example, OPERand).

For the z/OS UNIX commands, the keywords must be entered in the case indicated in the syntax diagram.

Variables are italicized, appear in lowercase letters, and represent names or values you supply. For example, a data set is a variable.

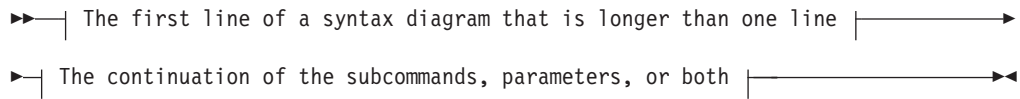
Syntax examples

In the following example, the USER command is a keyword. The required variable parameter is *user_id*, and the optional variable parameter is *password*. Replace the variable parameters with your own values.



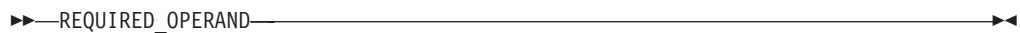
Longer than one line

If a diagram is longer than one line, the first line ends with a single arrowhead and the second line begins with a single arrowhead.



Required operands

Required operands and values appear on the main path line.



You must code required operands and values.

Choose one required item from a stack

If there is more than one mutually exclusive required operand or value to choose from, they are stacked vertically.



Optional values

Optional operands and values appear below the main path line.



You can choose not to code optional operands and values.

Choose one optional operand from a stack

If there is more than one mutually exclusive optional operand or value to choose from, they are stacked vertically below the main path line.



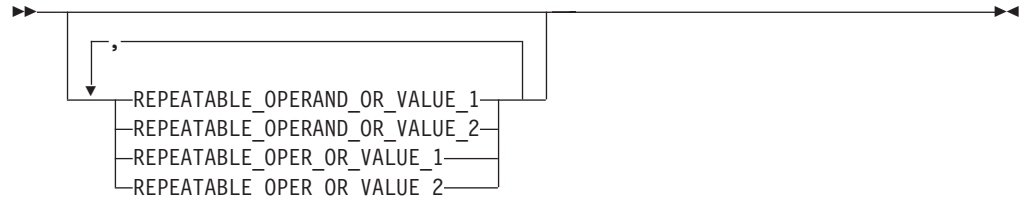
Repeating an operand

An arrow returning to the left above an operand or value on the main path line means that the operand or value can be repeated. The comma means that each operand or value must be separated from the next by a comma. If no comma appears in the returning arrow, the operand or value must be separated from the next by a blank.



Selecting more than one operand

An arrow returning to the left above a group of operands or values means more than one can be selected, or a single one can be repeated.



Nonalphanumeric characters

If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code OPERAND=(001,0.001).



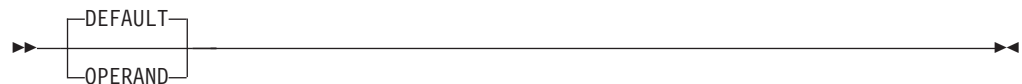
Blank spaces in syntax diagrams

If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code OPERAND=(001 FIXED).



Default operands

Default operands and values appear above the main path line. TCP/IP uses the default if you omit the operand entirely.



Variables

A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.



Syntax fragments

Some diagrams contain syntax fragments, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed

case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

►► | Syntax fragment | ◄◄

Syntax fragment:

|—1ST_OPERAND,2ND_OPERAND,3RD_OPERAND—|

Appendix D. Related protocol specifications (RFCs)

This appendix lists the related protocol specifications for TCP/IP. The Internet Protocol suite is still evolving through requests for comments (RFC). New protocols are being designed and implemented by researchers and are brought to the attention of the Internet community in the form of RFCs. Some of these protocols are so useful that they become recommended protocols. That is, all future implementations for TCP/IP are recommended to implement these particular functions or protocols. These become the *de facto* standards, on which the TCP/IP protocol suite is built.

These documents can be obtained from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

You can see Internet drafts at <http://www.ietf.org/ID.html>. See "Draft RFCs" on page 740 for draft RFCs implemented in z/OS V1R4 Communications Server.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to service@nic.ddn.mil with a subject line of RFC *nnnn* for text versions or a subject line of RFC *nnnn*.PS for PostScript versions. To request a copy of the RFC index, send a message with a subject line of RFC INDEX.

For more information, contact nic@nic.ddn.mil.

Many RFCs are available online. Hard copies of all RFCs are available from the NIC, either individually or by subscription. Online copies are available using FTP from the NIC at the following Web address: <http://www.rfc-editor.org/rfc.html>.

Use FTP to download the files, using the following format:

RFC:RFC-INDEX.TXT
RFC:RFC*nnnn*.TXT
RFC:RFC*nnnn*.PS

where:

nnnn Is the RFC number.
TXT Is the text format.
PS Is the PostScript format.

Many features of TCP/IP Services are based on the following RFCs:

RFC	Title and Author
768	<i>User Datagram Protocol</i> J.B. Postel
791	<i>Internet Protocol</i> J.B. Postel
792	<i>Internet Control Message Protocol</i> J.B. Postel
793	<i>Transmission Control Protocol</i> J.B. Postel
821	<i>Simple Mail Transfer Protocol</i> J.B. Postel

- 822 *Standard for the Format of ARPA Internet Text Messages* D. Crocker
- 823 *DARPA Internet Gateway* R.M. Hinden, A. Sheltzer
- 826 *Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.Bit Ethernet Address for Transmission on Ethernet Hardware* D.C. Plummer
- 854 *Telnet Protocol Specification* J.B. Postel, J.K. Reynolds
- 855 *Telnet Option Specification* J.B. Postel, J.K. Reynolds
- 856 *Telnet Binary Transmission* J.B. Postel, J.K. Reynolds
- 857 *Telnet Echo Option* J.B. Postel, J.K. Reynolds
- 858 *Telnet Suppress Go Ahead Option* J.B. Postel, J.K. Reynolds
- 859 *Telnet Status Option* J.B. Postel, J.K. Reynolds
- 860 *Telnet Timing Mark Option* J.B. Postel, J.K. Reynolds
- 861 *Telnet Extended Options—List Option* J.B. Postel, J.K. Reynolds
- 862 *Echo Protocol* J.B. Postel
- 863 *Discard Protocol* J.B. Postel
- 864 *Character Generator Protocol* J.B. Postel
- 877 *Standard for the Transmission of IP Datagrams over Public Data Networks* J.T. Korb
- 885 *Telnet End of Record Option* J.B. Postel
- 896 *Congestion Control in IP/TCP Internetworks* J. Nagle
- 903 *Reverse Address Resolution Protocol* R. Finlayson, T. Mann, J.C. Mogul, M. Theimer
- 904 *Exterior Gateway Protocol Formal Specification* D.L. Mills
- 919 *Broadcasting Internet Datagrams* J.C. Mogul
- 922 *Broadcasting Internet Datagrams in the Presence of Subnets* J.C. Mogul
- 950 *Internet Standard Subnetting Procedure* J.C. Mogul, J.B. Postel
- 952 *DoD Internet Host Table Specification* K. Harrenstien, M.K. Stahl, E.J. Feinler
- 959 *File Transfer Protocol* J.B. Postel, J.K. Reynolds
- 974 *Mail Routing and the Domain Name System* C. Partridge
- 1006 *ISO Transport Service on top of the TCP Version 3* M.T.Rose, D.E. Cass
- 1009 *Requirements for Internet Gateways* R.T. Braden, J.B. Postel
- 1011 *Official Internet Protocols* J. Reynolds, J. Postel
- 1013 *X Window System Protocol, Version 11: Alpha Update* R.W. Scheifler
- 1014 *XDR: External Data Representation Standard* Sun Microsystems Incorporated
- 1027 *Using ARP to Implement Transparent Subnet Gateways* S. Carl-Mitchell, J.S. Quarterman
- 1032 *Domain Administrators Guide* M.K. Stahl
- 1033 *Domain Administrators Operations Guide* M. Lottor

- 1034 *Domain Names—Concepts and Facilities* P.V. Mockapetris
- 1035 *Domain Names—Implementation and Specification* P.V. Mockapetris
- 1042 *Standard for the Transmission of IP Datagrams over IEEE 802 Networks*
J.B. Postel, J.K. Reynolds
- 1044 *Internet Protocol on Network System's HYPERchannel: Protocol
Specification* K. Hardwick, J. Lekashman
- 1055 *Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP* J.L.
Romkey
- 1057 *RPC: Remote Procedure Call Protocol Version 2 Specification* Sun
Microsystems Incorporated
- 1058 *Routing Information Protocol* C.L. Hedrick
- 1060 *Assigned Numbers* J. Reynolds, J. Postel
- 1073 *Telnet Window Size Option* D. Waitzman
- 1079 *Telnet Terminal Speed Option* C.L. Hedrick
- 1091 *Telnet Terminal-Type Option* J. VanBokkelen
- 1094 *NFS: Network File System Protocol Specification* Sun Microsystems
Incorporated
- 1096 *Telnet X Display Location Option* G. Marcy
- 1101 *DNS encoding of network names and other types* P.V. Mockapetris
- 1112 *Host Extensions for IP Multicasting* S. Deering
- 1118 *Hitchhikers Guide to the Internet* E. Krol
- 1122 *Requirements for Internet Hosts—Communication Layers* R.T. Braden
- 1123 *Requirements for Internet Hosts—Application and Support* R.T. Braden
- 1155 *Structure and Identification of Management Information for TCP/IP-Based
Internets* M.T. Rose, K. McCloghrie
- 1156 *Management Information Base for Network Management of TCP/IP-Based
Internets* K. McCloghrie, M.T. Rose
- 1157 *Simple Network Management Protocol (SNMP)* J.D. Case, M. Fedor, M.L.
Schoffstall, C. Davin
- 1158 *Management Information Base for Network Management of TCP/IP-based
internets: MIB-II* M.T. Rose
- 1179 *Line Printer Daemon Protocol* The Wollongong Group, L. McLaughlin III
- 1180 *TCP/IP Tutorial* T.J. Socolofsky, C.J. Kale
- 1183 *New DNS RR Definitions* C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V.
Mockapetris, (Updates RFC 1034, RFC 1035)
- 1184 *Telnet Linemode Option* D. Borman
- 1187 *Bulk Table Retrieval with the SNMP* M.T. Rose, K. McCloghrie, J.R. Davin
- 1188 *Proposed Standard for the Transmission of IP Datagrams over FDDI
Networks* D. Katz
- 1191 *Path MTU Discovery* J. Mogul, S. Deering
- 1198 *FYI on the X Window System* R.W. Scheifler

- 1207 *FYI on Questions and Answers: Answers to Commonly Asked "Experienced Internet User" Questions* G.S. Malkin, A.N. Marine, J.K. Reynolds
- 1208 *Glossary of Networking Terms* O.J. Jacobsen, D.C. Lynch
- 1213 *Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II* K. McCloghrie, M.T. Rose
- 1215 *Convention for Defining Traps for Use with the SNMP* M.T. Rose
- 1228 *SNMP-DPI Simple Network Management Protocol Distributed Program Interface* G.C. Carpenter, B. Wijnen
- 1229 *Extensions to the Generic-Interface MIB* K. McCloghrie
- 1230 *IEEE 802.4 Token Bus MIB* K. McCloghrie, R. Fox
- 1231 *IEEE 802.5 Token Ring MIB* K. McCloghrie, R. Fox, E. Decker
- 1236 *IP to X.121 Address Mapping for DDN* L. Morales, P. Hasse
- 1267 *A Border Gateway Protocol 3 (BGP-3)* K. Lougheed, Y. Rekhter
- 1268 *Application of the Border Gateway Protocol in the Internet* Y. Rekhter, P. Gross
- 1269 *Definitions of Managed Objects for the Border Gateway Protocol (Version 3)* S. Willis, J. Burruss
- 1270 *SNMP Communications Services* F. Kastenholz, ed.
- 1321 *The MD5 Message-Digest Algorithm* R. Rivest
- 1323 *TCP Extensions for High Performance* V. Jacobson, R. Braden, D. Borman
- 1325 *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* G.S. Malkin, A.N. Marine
- 1340 *Assigned Numbers* J.K. Reynolds, J.B. Postel
- 1348 *DNS NSAP RRs* B. Manning
- 1349 *Type of Service in the Internet Protocol Suite* P. Almquist
- 1350 *TFTP Protocol* K.R. Sollins
- 1351 *SNMP Administrative Model* J. Davin, J. Galvin, K. McCloghrie
- 1352 *SNMP Security Protocols* J. Galvin, K. McCloghrie, J. Davin
- 1353 *Definitions of Managed Objects for Administration of SNMP Parties* K. McCloghrie, J. Davin, J. Galvin
- 1354 *IP Forwarding Table MIB* F. Baker
- 1356 *Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode* A. Malis, D. Robinson, R. Ullmann
- 1363 *A Proposed Flow Specification* C. Partridge
- 1372 *Telnet Remote Flow Control Option* D. Borman, C. L. Hedrick
- 1374 *IP and ARP on HIPPI* J. Renwick, A. Nicholson
- 1381 *SNMP MIB Extension for X.25 LAPB* D. Throop, F. Baker
- 1382 *SNMP MIB Extension for the X.25 Packet Layer* D. Throop
- 1387 *RIP Version 2 Protocol Analysis* G. Malkin
- 1388 *RIP Version 2—Carrying Additional Information* G. Malkin

- 1389 *RIP Version 2 MIB Extension* G. Malkin
- 1390 *Transmission of IP and ARP over FDDI Networks* D. Katz
- 1393 *Traceroute Using an IP Option* G. Malkin
- 1397 *Default Route Advertisement In BGP2 And BGP3 Versions of the Border Gateway Protocol* D. Haskin
- 1398 *Definitions of Managed Objects for the Ethernet-Like Interface Types* F. Kastenholz
- 1416 *Telnet Authentication Option* D. Borman, ed.
- 1464 *Using the Domain Name System to Store Arbitrary String Attributes* R. Rosenbaum
- 1469 *IP Multicast over Token-Ring Local Area Networks* T. Pusateri
- 1535 *A Security Problem and Proposed Correction With Widely Deployed DNS Software* E. Gavron
- 1536 *Common DNS Implementation Errors and Suggested Fixes* A. Kumar, J. Postel, C. Neuman, P. Danzig, S. Miller
- 1537 *Common DNS Data File Configuration Errors* P. Beertema
- 1540 *IAB Official Protocol Standards* J.B. Postel
- 1571 *Telnet Environment Option Interoperability Issues* D. Borman
- 1572 *Telnet Environment Option* S. Alexander
- 1577 *Classical IP and ARP over ATM* M. Laubach
- 1583 *OSPF Version 2* J. Moy
- 1591 *Domain Name System Structure and Delegation* J. Postel
- 1592 *Simple Network Management Protocol Distributed Protocol Interface Version 2.0* B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, G. Waters
- 1594 *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* A.N. Marine, J. Reynolds, G.S. Malkin
- 1695 *Definitions of Managed Objects for ATM Management Version 8.0 Using SMIv2* M. Ahmed, K. Tesink
- 1706 *DNS NSAP Resource Records* B. Manning, R. Colella
- 1713 *Tools for DNS debugging* A. Romao
- 1723 *RIP Version 2—Carrying Additional Information* G. Malkin
- 1766 *Tags for the Identification of Languages* H. Alvestrand
- 1794 *DNS Support for Load Balancing* T. Brisco
- 1832 *XDR: External Data Representation Standard* R. Srinivasan
- 1850 *OSPF Version 2 Management Information Base* F. Baker, R. Coltun
- 1876 *A Means for Expressing Location Information in the Domain Name System* C. Davis, P. Vixie, T. Goodwin, I. Dickinson
- 1886 *DNS Extensions to support IP version 6* S. Thomson, C. Huitema
- 1901 *Introduction to Community-Based SNMPv2* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

- 1902 *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
- 1903 *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
- 1904 *Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
- 1905 *Protocols Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
- 1906 *Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
- 1907 *Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
- 1908 *Coexistence between Version 1 and Version 2 of the Internet-Standard Network Management Framework* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
- 1912 *Common DNS Operational and Configuration Errors* D. Barr
- 1918 *Address Allocation for Private Internets* Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot, E. Lear
- 1928 *SOCKS Protocol Version 5* M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones
- 1939 *Post Office Protocol-Version 3* J. Myers, M. Rose
- 1981 *Path MTU Discovery for IP version 6* J. McCann, S. Deering, J. Mogul
- 1982 *Serial Number Arithmetic* R. Elz, R. Bush
- 1995 *Incremental Zone Transfer in DNS* M. Ohta
- 1996 *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)* P. Vixie
- 2010 *Operational Criteria for Root Name Servers* B. Manning, P. Vixie
- 2011 *SNMPv2 Management Information Base for the Internet Protocol Using SMIv2* K. McCloghrie
- 2012 *SNMPv2 Management Information Base for the Transmission Control Protocol Using SMIv2* K. McCloghrie
- 2013 *SNMPv2 Management Information Base for the User Datagram Protocol Using SMIv2* K. McCloghrie
- 2052 *A DNS RR for specifying the location of services (DNS SRV)* A. Gulbrandsen, P. Vixie
- 2065 *Domain Name System Security Extensions* D. Eastlake, C. Kaufman
- 2096 *IP Forwarding Table MIB* F. Baker
- 2104 *HMAC: Keyed-Hashing for Message Authentication* H. Krawczyk, M. Bellare, R. Canetti
- 2132 *DHCP Options and BOOTP Vendor Extensions* S. Alexander, R. Droms
- 2133 *Basic Socket Interface Extensions for IPv6* R. Gilligan, S. Thomson, J. Bound, W. Stevens

- 2137 *Secure Domain Name System Dynamic Update* D. Eastlake
- 2163 *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)* C. Allocchio
- 2168 *Resolution of Uniform Resource Identifiers using the Domain Name System* R. Daniel, M. Mealling
- 2178 *OSPF Version 2* J. Moy
- 2181 *Clarifications to the DNS Specification* R. Elz, R. Bush
- 2205 *Resource ReSerVation Protocol (RSVP) Version 1* R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin
- 2210 *The Use of RSVP with IETF Integrated Services* J. Wroclawski
- 2211 *Specification of the Controlled-Load Network Element Service* J. Wroclawski
- 2212 *Specification of Guaranteed Quality of Service* S. Shenker, C. Partridge, R. Guerin
- 2215 *General Characterization Parameters for Integrated Service Network Elements* S. Shenker, J. Wroclawski
- 2219 *Use of DNS Aliases for Network Services* M. Hamilton, R. Wright
- 2228 *FTP Security Extensions* M. Horowitz, S. Lunt
- 2230 *Key Exchange Delegation Record for the DNS* R. Atkinson
- 2233 *The Interfaces Group MIB Using SMIv2* K. McCloghrie, F. Kastenholz
- 2240 *A Legal Basis for Domain Name Allocation* O. Vaughn
- 2246 *The TLS Protocol Version 1.0* T. Dierks, C. Allen
- 2308 *Negative Caching of DNS Queries (DNS NCACHE)* M. Andrews
- 2317 *Classless IN-ADDR.ARPA delegation* H. Eidnes, G. de Groot, P. Vixie
- 2320 *Definitions of Managed Objects for Classical IP and ARP over ATM Using SMIv2* M. Greene, J. Luciani, K. White, T. Kuo
- 2328 *OSPF Version 2* J. Moy
- 2345 *Domain Names and Company Name Retrieval* J. Klensin, T. Wolf, G. Oglesby
- 2352 *A Convention for Using Legal Names as Domain Names* O. Vaughn
- 2355 *TN3270 Enhancements* B. Kelly
- 2373 *IP Version 6 Addressing Architecture* R. Hinden, M. O'Dell, S. Deering
- 2374 *An IPv6 Aggregatable Global Unicast Address Format* R. Hinden, M. O'Dell, S. Deering
- 2375 *IPv6 Multicast Address Assignments* R. Hinden, S. Deering
- 2389 *Feature negotiation mechanism for the File Transfer Protocol* P. Hethmon, R. Elz
- 2428 *FTP Extensions for IPv6 and NATs* M. Allman, S. Ostermann, C. Metz
- 2460 *Internet Protocol, Version 6 (IPv6) Specification* S. Deering, R. Hinden
- 2461 *Neighbor Discovery for IP Version 6 (IPv6)* T. Narten, E. Nordmark, W. Simpson
- 2462 *IPv6 Stateless Address Autoconfiguration* S. Thomson, T. Narten

- 2464** *Transmission of IPv6 Packets over Ethernet Networks* M. Crawford
- 2474** *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers* K. Nichols, S. Blake, F. Baker, D. Black
- 2535** *Domain Name System Security Extensions* D. Eastlake
- 2539** *Storage of Diffie-Hellman Keys in the Domain Name System (DNS)* D. Eastlake
- 2553** *Basic Socket Interface Extensions for IPv6* R. Gilligan, S. Thomson, J. Bound, W. Stevens
- 2571** *An Architecture for Describing SNMP Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen
- 2572** *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen
- 2573** *SNMP Applications* D. Levi, P. Meyer, B. Stewart
- 2574** *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen
- 2575** *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie
- 2578** *Structure of Management Information Version 2 (SMIv2)* K. McCloghrie, D. Perkins, J. Schoenwaelder
- 2640** *Internationalization of the File Transfer Protocol* B. Curtin
- 2665** *Definitions of Managed Objects for the Ethernet-like Interface Types* J. Flick, J. Johnson
- 2672** *Non-Terminal DNS Name Redirection* M. Crawford
- 2710** *Multicast Listener Discovery (MLD) for IPv6* S. Deering, W. Fenner, B. Haberman
- 2711** *IPv6 Router Alert Option* C. Partridge, A. Jackson
- 2758** *Definitions of Managed Objects for Service Level Agreements Performance Monitoring* K. White
- 2845** *Secret Key Transaction Authentication for DNS (TSIG)* P. Vixie, O. Gudmundsson, D. Eastlake, B. Wellington
- 2874** *DNS Extensions to Support IPv6 Address Aggregation and Renumbering* M. Crawford, C. Huitema
- 2941** *Telnet Authentication Option* T. Ts'o, ed., J. Altman
- 2942** *Telnet Authentication: Kerberos Version 5* T. Ts'o
- 2946** *Telnet Data Encryption Option* T. Ts'o
- 2952** *Telnet Encryption: DES 64 bit Cipher Feedback* T. Ts'o
- 2953** *Telnet Encryption: DES 64 bit Output Feedback* T. Ts'o, ed.
- 3060** *Policy Core Information Model—Version 1 Specification* B. Moore, E. Ellesson, J. Strassner, A. Westerinen

Draft RFCs

Several areas of IPv6 implementation include elements of the following draft RFCs and are subject to change during the RFC review process.

| **Advanced Sockets API for IPv6**

| W. Richard Stevens, Matt Thomas, Erik Nordmark, Tatuya Jinmei

| **Basic Socket Interface Extensions for IPv6**

| R.E. Gilligan, S. Thomson, J. Bound, J. McCann, W. R. Stevens

| **Default Address Selection for IPv6**

| R. Draves

| **Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version
6 (IPv6) Specification**

| A. Conta, S. Deering

| **IP Version 6 Addressing Architecture**

| R. Hinden, S. Deering

Appendix E. Information APARs

This appendix lists information APARs for IP and SNA documents.

Notes:

1. Information APARs contain updates to previous editions of the manuals listed below. Documents updated for V1R4 are complete except for the updates contained in the information APARs that may be issued after V1R4 documents went to press.
2. Information APARs are predefined for z/OS V1R4 Communications Server and may not contain updates.
3. Information APARs for OS/390 documents are in the document called *OS/390 DOC APAR and PTF ++HOLD Documentation*, which can be found at http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IDDOCMST/CCONTENTS.
4. Information APARs for z/OS documents are in the document called *z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation*, which can be found at http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS.

Information APARs for IP documents

Table 61 lists information APARs for IP documents.

Table 61. IP information APARs

Title	z/OS CS V1R4	z/OS CS V1R2	CS for OS/390 2.10 and z/OS CS V1R1	CS for OS/390 2.8
IP API Guide	ii13255	ii12861	ii12371	ii11635
IP CICS Sockets Guide	ii13257	ii12862		ii11626
IP Configuration				ii11620 ii12068 ii12353 ii12649 ii13018
IP Configuration Guide	ii13244	ii12498 ii13087	ii12362 ii12493 ii13006	
IP Configuration Reference	ii13245	ii12499	ii12363 ii12494 ii12712	
IP Diagnosis	ii13249	ii12503	ii12366 ii12495	ii11628
IP Messages Volume 1	ii13250	ii12857 ii13229	ii12367	ii11630 13230
IP Messages Volume 2	ii13251	ii12858	ii12368	ii11631
IP Messages Volume 3	ii13252	ii12859	ii12369 12990	ii11632 ii12883
IP Messages Volume 4	ii13253	ii12860		
IP Migration	ii13242	ii12497	ii12361	ii11618

Table 61. IP information APARs (continued)

Title	z/OS CS V1R4	z/OS CS V1R2	CS for OS/390 2.10 and z/OS CS V1R1	CS for OS/390 2.8
IP Network and Application Design Guide	ii13243			
IP Network Print Facility		ii12864		ii11627
IP Programmer's Reference	ii13256	ii12505		ii11634
IP and SNA Codes	ii13254	ii12504	ii12370	ii11917
IP User's Guide			ii12365 ii13060	ii11625
IP User's Guide and Commands	ii13247	ii12501	ii12365 ii13060	ii11625
IP System Admin Guide	ii13248	ii12502		
Quick Reference	ii13246	ii12500	ii12364	

Information APARs for SNA documents

Table 62 lists information APARs for SNA documents.

Table 62. SNA information APARs

Title	z/OS CS V1R4	z/OS CS V1R2	CS for OS/390 2.10 and z/OS CS V1R1	CS for OS/390 2.8
Anynet SNA over TCP/IP				ii11922
Anynet Sockets over SNA				ii11921
CSM Guide				
IP and SNA Codes	ii13254	ii12504	ii12370	ii11917
SNA Customization	ii13240	ii12872	ii12388	ii11923
SNA Diagnosis	ii13236	ii12490 ii13034	ii12389	ii11915
SNA Messages	ii13238	ii12491	ii12382 ii12383	ii11916
SNA Network Implementation Guide	ii13234	ii12487	ii12381	ii11911
SNA Operation	ii13237	ii12489	ii12384	ii11914
SNA Migration	ii13233	ii12486	ii12386	ii11910
SNA Programming	ii13241	ii13033	ii12385	ii11920
Quick Reference	ii13246	ii12500	ii12364	ii11913
SNA Resource Definition Reference	ii13235	ii12488	ii12380 ii12567	ii11912 ii12568
SNA Resource Definition Samples				
SNA Data Areas	ii13239	ii12492	ii12387	ii11617

Other information APARs

Table 63 on page 745 lists information APARs not related to documents.

Table 63. Non-document information APARs

Content	Number
OMPROUTE	ii12026
iQDIO	ii11220
index of recommended maintenance for VTAM	ii11220
CSM for VTAM	ii12657
CSM for TCP/IP	ii12658
AHHC, MPC, and CTC	ii01501
DLUR/DLUS for z/OS V1R2	ii12986
Enterprise Extender	ii12223
Generic resources	ii10986
HPR	ii10953
MNPS	ii10370
Performance	ii11710 ii11711 ii11712

Appendix F. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Notices

IBM may not offer all of the products, services, or features discussed in this document. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O.Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly

tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

This product includes cryptographic software written by Eric Young.

If you are viewing this information softcopy, photographs and color illustrations may not appear.

You can obtain softcopy from the z/OS Collection (SK3T-4269), which contains BookManager and PDF formats of unlicensed books and the z/OS Licensed Product Library (LK3T-4307), which contains BookManager and PDF formats of licensed books.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM	Micro Channel
Advanced Peer-to-Peer Networking	MVS
AFP	MVS/DFP
AD/Cycle	MVS/ESA
AIX	MVS/SP
AIX/ESA	MVS/XA
AnyNet	MQ
APL2	Natural
AS/400	NetView
AT	Network Station
BookManager	Nways
BookMaster	Notes
CBPDO	NTune
C/370	NTuneNCP
CICS	OfficeVision/MVS
CICS/ESA	OfficeVision/VM
C/MVS	Open Class
Common User Access	OpenEdition
C Set ++	OS/2
CT	OS/390
CUA	OS/400
DATABASE 2	Parallel Sysplex
DatagLANce	Personal System/2
DB2	PR/SM
DFSMS	PROFS
DFSMSdfp	PS/2
DFSMSHsm	RACF
DFSMS/MVS	Resource Link
DPI	Resource Measurement Facility
Domino	RETAIN
DRDA	RFM
eNetwork	RISC System/6000
Enterprise Systems Architecture/370	RMF
ESA/390	RS/6000
ESCON	S/370
eServer	S/390
ES/3090	SAA
ES/9000	SecureWay
ES/9370	Slate
EtherStreamer	SP
Extended Services	SP2
FAA	SQL/DS
	System/360

FFST	System/370
FFST/2	System/390
FFST/MVS	SystemView
First Failure Support Technology	Tivoli
GDDM	TURBOWAYS
Hardware Configuration Definition	UNIX System Services
IBM	Virtual Machine/Extended Architecture
IBMLink	VM/ESA
IBMLINK	VM/XA
IMS	VSE/ESA
IMS/ESA	VTAM
InfoPrint	WebSphere
Language Environment	XT
LANStreamer	z/Architecture
Library Reader	z/OS
LPDA	z/OS.e
MCS	zSeries
	400
	3090
	3890

Lotus, Freelance, and Word Pro are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

DB2 and NetView are registered trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the U.S., other countries, or both.

The following terms are trademarks of other companies:

ATM is a trademark of Adobe Systems, Incorporated.

BSC is a trademark of BusiSoft Corporation.

CSA is a trademark of Canadian Standards Association.

DCE is a trademark of The Open Software Foundation.

HYPERchannel is a trademark of Network Systems Corporation.

UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see <http://www.intel.com/sites/corporate/tradmarx.htm> .

Other company, product, and service names may be trademarks or service marks of others.

Index

A

ABEND dumps 12
abends
 CEEDUMP 21
 dumps of ASIDs 56
 FTP client 341
 FTP server 314
 LPD (line printer daemon) 285
 NCPROUTE 622
 OMPROUTE 603
 OROUTED 583
 platform 21
 SMTP (simple mail transfer protocol) 384
 SNALINK LU0 404
 SNALINK LU6.2 439
 SNMP (simple network management protocol) 491
 SNTP (simple network time protocol) 699
 system 21
 tn3270 Telnet client 374
 tn3270 Telnet server 366
 user 21
accessibility features 747
active routes versus static routes 581, 719
ADDRBLOK data set 388
ADDRESS value, verifying 30
addressing on the Internet 719
allocation tables, DU 195
AMBLIST 16
APAR (authorized program analysis report) 5
API, TCPIPICS subcommand 173
ARP cache, querying 38
authorized program analysis report (APAR) 5
autolog, FTP server problems with 314

B

bridge 713, 715, 721

C

C_INET PFS
 OMPROUTE behavior in 603
 OROUTED 583
CEEDUMP
 description 21
 for SNMP abends 491
CICS (customer information control system) 673
CICS, location of trace output 11
Class D group addresses 719
common INET
 OMPROUTE behavior in 603
 OROUTED 583
common storage tracking 17
Communications Server for z/OS, online
 information xix
component ID, TCP/IP 19, 20

component trace
 displaying status 55
 general description 13
 managed data set 8
 OMPROUTE 601
 stopping 56
 tn3270 Telnet 373
 use with DDNS (dynamic domain name server) 452
 use with OMPROUTE, see also OMPROUTE
 application trace 615
 viewing data without an abend 56
component trace, changing options
 with PARMLIB member 53
 without PARMLIB member 54
CONFIG, TCPIPICS subcommand 175
configuration values, displaying device 32
connection optimization
 addresses not returned 452
 connection problems 453
CONNECTION, TCPIPICS subcommand 177
console messages, DDNS name server 443
control block addresses, displaying TCP/IP 227
control blocks, CICS sockets 676
control blocks, displaying
 MTCB (master TCP control block) 241
 MUCB (master UDP control block) 256
 socket 225
 stream 239
 TCBs (TCP control blocks) 241
 Telnet 243
 timer 244
 UCBs (UDP control blocks) 256
COUNTERS, TCPIPICS subcommand 178
CTRACE, IPCS subcommand 58
CTRACE, resolver 695
customer information control system (CICS) 673
customer number 19

D

data block, displaying 273
data buffers, displaying 273
data link control (DLC) 422
data traces
 displaying 135
 general description 13
 starting 135
date and time formatting 276
DATTRACE command 135
DB2
 interaction with NDB 467
 query support for FTP client 345
 query support for FTP server 323
DBCS (double-byte character set)
 FTP client 336
 FTP server 311
DDNS (dynamic domain name server) 441

- DDNS problems
 - connection optimization 452
 - console messages 443
 - debug option 445
 - name server signals 448
 - nsupdate 451
 - onslookup and nslookup 445
 - resolver directive 447
 - return codes 451
 - syslog messages 443
 - TSO DIG 445
- DDNS, location of trace output 9
- debug traces
 - for SNALINK LU0 413
 - for z/OS UNIX applications 14
 - SNTP (simple network time protocol) 699
- debugging switches, z/OS UNIX sendmail 393
- device interface, connecting 177, 178
- device interface, displaying 58, 175
- diagnosing resolver problems
 - resolving the hostname
 - steps for 683
- disability, physical 747
- dispatchable unit control block (DUCB), displaying 192
- dispatchable units, displaying TCP/IP 227
- DISPLAY command, use with LUs 431
- display commands 16
- DISPLAY TCPIP,STOR 16
- DISPLAY TRACE command 55
- distributed protocol interface (DPI) 487, 620
- DLC (data link control) 422
- DNS, online information xx
- documents, licensed xx
- domain name server, dynamic (DDNS) 441
- double-byte character set (DBCS)
 - FTP client 336
 - FTP server 311
- DPI (distributed protocol interface) 487, 620
- DU allocation tables, displaying 195
- DUAF, TCPIPICS subcommand 192
- DUCB, displaying 192
- DUCB, TCPIPICS subcommand 195
- DUMP command 56
- dumps
 - ABEND 7
 - definition 7, 21
 - FFST 13, 15
 - stand-alone 13
 - SVC 8, 13
 - SYSABEND 12
 - SYSMDUMP 12
 - SYSUDUMP 12
 - to obtain component trace data 56

E

- errno reason code, displaying 262
- ERRNO, IPCS subcommand 262
- errnojr reason code, displaying 262
- error exit codes, FTP server 310

- Ethernet
 - defining for NCPROUTE 621
 - DIX V2 717
- Express Logon, location of trace output 11
- EZAFTPAP start procedure 310

F

- FDDI (fiber distributed data interface) 715
- FFST (first failure support technology)
 - console 16
 - dump 13, 15
 - general description 14
 - minidump 15
 - probes 703, 704
 - sdump 15
- fiber distributed data interface (FDDI) 715
- field maintenance ID 20
- file tagging support
 - FTP client 338
- file tagging support, FTP client
 - ASCII file transfers 338
 - binary file transfers 340
- file transfer protocol (FTP), see also FTP 309
- FIREWALL, TCPIPICS subcommand 200
- first failure support technology (FFST) 14
- first failure support technology (FFST), see also FFST 703
- FRCA, TCPIPICS subcommand 203
- FTP client
 - customizing configuration parameters 336
 - DB2 345
 - DBCS (double-byte character set) 336
 - file tagging support 338
 - MBCS (mult-byte character set) 337
 - naming HFS files 311
 - naming MVS data sets 310
 - overview 335
 - SBCS (single-byte character set) 336
 - translation and data conversion support 336
 - using FTP.DATA 336
 - using TCPIP.DATA 336
 - using traces 347
- FTP client problems
 - "Unknown Host" message 341
 - abends 341
 - data transfer 342
 - documentation for IBM Support Center 348
- FTP server
 - customizing server parameters 310
 - customizing the start procedure 310
 - DB2 query support 323
 - DBCS (double-byte character set) 311
 - error exit codes 310
 - LOADLIB directory information 322
 - location of trace output 8
 - MBCS (multibyte character set) 312
 - naming HFS files 311
 - naming MVS data sets 310
 - overview 309
 - role of TCPIP.DATA 310

- FTP server (*continued*)
 - SBCS (single-byte character set) 312
 - traces 326
 - translation and data conversion support 311
- FTP server problems
 - abends 314
 - autolog 314
 - checkpoint markers 322
 - client abends during RETR 322
 - command failure 321
 - configuration values 313
 - connection 314, 315, 316
 - data set allocation 319
 - data set disposition 322
 - documentation for IBM Support Center 335
 - incomplete initialization 313
 - job entry subsystem (JES) 325
 - logon failure 315
 - lost messages and traces 326
 - MVS data set 320
 - password validation fails 315
 - port specification 314
 - SQL 323
 - terminated data transfer 321
 - user exit routine 326
 - working directory 316
- FTP.DATA data set
 - definition 310
 - specifying working directory 316
- FTP.DATA, TRACE statement in 327

G

- gateway (router)
 - broadcasting routing tables 581
 - definition 713
 - identifying configured gateways 584
 - indirect routing 723
 - internet addressing 721
 - LPR and LPD problems 286
 - using PING/oping to check connectivity 27, 29
- gateway IP address 224
- GATEWAY statement
 - configuring routing tables 727
 - NCPROUTE GATEWAYS data set 630
 - problems uncovered by NETSTAT/onetstat 37
 - problems with X.25 644, 646, 650
 - subnet masks 724
 - use with NCPROUTE 623, 624
 - use with OMPROUTE 602
- global work area (GWA) 676
- GWA (global work area) 676

H

- hangs, collecting documentation 23
- hardware problem 11
- hash tables, displaying 205
- HASH, TCPIPICS subcommand 205
- header fields, displaying ICMPP 264
- header fields, displaying IP 265

- header fields, displaying TCP 274
- header, displaying Resolver 267
- header, displaying UDP 277
- header, system dump 209
- HEADER, TCPIPICS subcommand 209
- HELP, TCPIPICS subcommand 210
- HFS (hierarchical file system)
 - file names 25
 - file naming conventions for FTP 311
 - required for OMPROUTE 601

I

- IBM Software Support Center, contacting xxi
- IBM Software Support Center, documentation for
 - FTP client problems 348
 - FTP server problems 335
 - LPD (line printer daemon) problems 286
 - SNMP problems 492, 498
 - X.25 NPSI 651
- IBM Software Support Center, when to call 5
- IBM Support Center, documentation for
 - general information 19
 - server connection problems 49
- ICMPP header fields, displaying 264
- ICMPPHDR, IPCS subcommand 264
- IEEE 802.3 717
- IEEE 802.5, relationship between RC and I fields 716
- IMS socket interface
 - configuration 654
 - NETSTAT command with 667
 - overview 653
- IMS socket interface problems
 - bad connections 658
 - building a component 663
 - configuration mistakes 656
 - data transfer 661
 - error messages and return codes 661, 669
 - starting and stopping components 657
 - unexpected database actions 664
- IMS socket interface traces 666, 667
- IMS, location of trace output 11
- inetd.conf file 463
- information APARs for IP-related documents 743
- information APARs for non- document information 744
- information APARs for SNA-related documents 744
- interactive problem control system, see also IPCS 17
- interface, connecting 177, 178
- interface, displaying 58, 175
- Internet
 - addressing 719
 - protocol (IP) 719
- Internet, finding z/OS information online xix
- IP header fields, displaying 265
- IPCS (interactive problem control system)
 - definition 17
 - sending print output 272
- IPCS subcommand
 - CTRACE 58
 - ERRNO 262
 - RESOLVER 267

IPCS subcommand (*continued*)
 SETPRINT 272
 SKMSG 273
 TCPHDR 274
 TCIPCS 171
 TOD 276
 UDPHDR 277
 IPHDR, IPCS subcommand 265
 IPv4 subnetting 724

J

JES (job entry subsystem), output not found 325
 job entry subsystem (JES), output not found 325

K

keyboard 747

L

LE runtime library 19
 license, patent, and copyright information 749
 licensed documents xx
 line printer daemon, see also LPD 285
 line printer requester, see also LPR 285
 LINK value, verifying 30
 LIST MODIFY command 430
 load modules 11, 16
 LOCK, TCIPCS subcommand 211
 locks, displaying 211
 LOCKSUM option 211
 logon problems, tn3270 Telnet server 366
 LOGREC data set 17
 loops
 collecting documentation 22
 definition 22
 SMTP 386
 LPD (line printer daemon)
 abends 285
 activating server traces 294
 DEBUG option 295, 302
 overview 285
 timeouts, hangs, waits 286
 unsupported filter 300
 LPD server, location of trace output 8
 LPR (line printer requester)
 activating client traces 288
 creating client traces 288
 FILTER X option 291
 garbled data 287
 identifying a port 292
 missing data 287
 nonworking options 288
 overview 285
 timeouts, hangs, waits 286
 XLATE option 293
 LPR client, location of trace output 8

M

management information base (MIB) 487, 621
 MAP, TCIPCS subcommand 213
 master TCB control block, displaying 241
 master trace 14
 master UDP control block (MUCB), displaying 256
 maximum transmission unit (MTU) 714
 MBCS (multibyte character set)
 FTP client 337
 FTP server 312
 message block, displaying the stream 273
 messages, OROUTED 582
 MIB (management information base) 487, 621
 MODIFY command
 with NCPROUTE 629
 with OROUTED 588, 589, 590, 591
 module ID, displaying 262
 module tables, displaying 214
 motif 483
 MTABLE, TCIPCS subcommand 214
 MTU (maximum transmission unit) 714
 MUCB (master UDP control block), displaying 256
 multibyte character set (MBCS)
 FTP client 337
 FTP server 312
 multicast, Class D addresses 719
 multiple stacks 25, 58, 494, 583, 603
 MVS data set
 FTP naming conventions 310
 FTP server unable to find 320
 MVS system console log for REXECD 459

N

name server signals 448
 name server, diagnosing problems with 441
 named command 445
 NCPROUTE
 abends 622
 activating global traces 629
 activating selective traces 630
 bad connections 622
 communication with SNMP 620, 621
 defining 621
 incorrect output 627
 overview 619
 PING failure 625
 RIP (routing information protocol) commands 620
 session outages 628
 trace example 631
 NCRROUTE, location of trace output 10
 NDB (network database)
 definitions required 468
 documentation required for problems 468, 469
 obtaining DB2 data 469
 overview 467
 port manager trace 470, 472
 NDB, location of trace output 9
 NDBC control block 469
 netdata, z/OS UNIX Telnet 350

- NETSTAT command
 - diagnosing a timeout 29
 - displaying data trace 135
 - verifying SNALINK LU6.2 429
 - with IMS socket interface 666, 667
 - with SNALINK LU0 405
- NetView 15, 488, 489, 492, 627
- network database, see also NDB 467
- network IP addresses 719
- network, problems with SNA 424, 426
- NSLOOKUP command 445
- nsupdate command 451

O

- OMPROUTE
 - abends 603
 - client cannot reach destination 603
 - component trace support for 615
 - connection problems 603
 - overview 601
- OMPROUTE subagent
 - start options 490
 - statements 490
- onetstat command
 - diagnosing a timeout 29
 - displaying device configuration values 32
 - displaying routing information 27, 36, 37
 - OMPROUTE problem diagnosis 603
 - port determination 27
 - querying ARP cache 38
 - use with data trace 135
 - verifying ADDRESS and LINK values 30
 - verifying device status 28
 - viewing host addresses 25
- onslookup command 445
- open shortest path first (OSPF) 601
- open software foundation (OSF) 483
- operating system identification 19
- oping command
 - checking connectivity 27
 - compared to TSO PING command 28
 - timeout 29
 - using 28
 - verifying device connection 29
 - verifying network definition 29
 - verifying onetstat -h 28
 - verifying packets sent and received 29
 - verifying route back to local host 29
 - verifying route definition 29
 - verifying router can forward 29
 - verifying TCP/IP 28
- options debug directive 447
- orexec (remote execution protocol), z/OS UNIX
 - trace example 464
- orexecd (remote execution protocol daemon), z/OS UNIX, activating debug trace 465
- OROUTED
 - location of trace output 10
 - overview 581
 - starting traces 588, 589

- OROUTED (*continued*)
 - stopping 590
 - trace output 590
- OROUTED problems
 - abends 583
 - analyzing routing failures 584
 - connection 583
 - incorrect output 586
 - oping/ping failures 584
 - session outages 586
 - trace example 591
- OROUTED, interaction with
 - onetstat/netstat command 584
 - oping/ping command 584
- orshd (remote shell daemon), z/OS UNIX
 - activating debug trace 466
- OSA-Express 39
- OSF (open software foundation) 483
- OSF/Motif, location of trace output 10
- osnmp command 488, 489, 507
- OSPF (open shortest path first) 601
- otelnet, diagnostic messages sent to wrong file 349
- otracert command 40

P

- packet trace
 - general description 14, 92
 - starting 93
 - with IMS socket interface 666
 - with SNALINK LU0 406
 - with SNALINK LU6.2 428, 437
- patricia trees, displaying 250
- PDUs (protocol data units)
 - NCPROUTE 620
 - SNMP (simple network management protocol) 487
- PFS (physical file structure) 135
- physical file structure (PFS) 135
- PING command
 - compared to z/OS UNIX oping command 28
 - NCPROUTE client problems 625
 - SNMP (simple network management protocol) 41
 - using 28
 - verifying device connection 29
 - verifying network definition 29
 - verifying onetstat -h 28
 - verifying packets sent and received 29
 - verifying route back to local host 29
 - verifying route definition 29
 - verifying router can forward 29
 - verifying TCP/IP 28
 - with IMS socket interface 656
 - X.25 NPSI 650
- PING timeout 29
- PKTTRACE statement 92
- platform abends 21
- policies 563
- policy agent problems
 - gathering diagnostic information 532
 - initialization problems 534
 - LDAP problems 537

- policy agent problems *(continued)*
 - log file example 541
 - overview 531
 - policy definition problems 535
 - service policy scopes 531
- policy agent terms
 - differentiated services 531
 - integrated services 531
 - quality of service (QoS) 531
 - resource reservation protocol (RSVP) 531
 - service differentiation 531
 - service level agreement (SLA) 531
 - service policy 531
- Policy Agent, location of trace output 10
- POLICY, TCPIPICS subcommand 215
- popper, diagnostic aids for z/OS UNIX 400
- Popper, location of trace output 9
- print output, sending IPCS 272
- problem number 19
- Problem reporting
 - machine-readable documentation 18
- PROFILE, TCPIPICS subcommand 217
- program objects 16
- protocol data units (PDUs)
 - NCPROUTE 620
 - SNMP (simple network management protocol) 487
- PROTOCOL, TCPIPICS subcommand 219
- ptydata, z/OS UNIX Telnet 350

R

- RACF (resource access control facility), VARY
 - command and 135
- RAW, TCPIPICS subcommand 222
- reason codes, displaying 262
- register save area (RSA) 192
- release number, TCP/IP 19
- remote execution protocol daemon, see also REXECD 455, 463
- remote execution protocol, see also REXEC 455, 463
- remote shell client, see also RSH 455
- remote shell daemon, see also RSH 463
- Resolver 683
- Resolver header, displaying 267
- RESOLVER trace 27, 390
- resolver, debug 447
- RESOLVER, IPCS subcommand 267
- Resolver, location of trace output 11
- resource access control facility (RACF), VARY
 - command and 135
- resource measurement facility 17
- RETAIN database 5
- return codes, DDNS name server 451
- REXEC (remote execution protocol), non-z/OS UNIX
 - activating debug trace 456
 - overview 455
 - problem documentation 455
 - trace example 456
- REXEC (remote execution protocol), z/OS UNIX
 - debug trace 464
 - trace example 464

- REXECD (remote execution protocol daemon), non-z/OS UNIX
 - command options for tracing 459
 - overview 459
 - problem documentation 459
 - trace example 460
- REXECD (remote execution protocol daemon), z/OS UNIX
 - trace example 465
- REXECD (remote execution protocol daemon), z/OS UNIX, activating debug trace 465
- REXX executables for TCPIPICS 171
- RFC (request for comment)
 - list of 733
- RFC (request for comments)
 - accessing online xix
- RIP (routing information protocol)
 - NCPROUTE implementation 619, 627
 - OMPROUTE implementation 601
 - OROUTED implementation 581, 727
- ROUTE, TCPIPICS subcommand 224
- router (gateway)
 - broadcasting routing tables 581
 - definition 713
 - identifying configured gateways 584
 - indirect routing 723
 - internet addressing 721
 - LPR and LPD problems 286
 - using PING/oping to check connectivity 27, 29
- routing
 - datagram algorithm 723
 - datagram algorithm with subnets 726
 - direct 722
 - dynamic 727
 - indirect 723
 - static 727
- routing control blocks, displaying 224
- routing information protocol (RIP)
 - NCPROUTE implementation 619, 627
 - OMPROUTE implementation 601
 - OROUTED implementation 581
- routing information, displaying 37
- routing table entry, displaying 224
- routing tables, displaying 36
- RSA (register save area) 192
- RSH (remote shell client), non-z/OS UNIX
 - client trace using SEND command 460
 - overview 455
 - trace example 458
- RSHD (remote shell daemon), z/OS UNIX
 - activating debug trace 466
 - random errors 466
 - trace example 466
- RSVP agent problems
 - application problems 565
 - gathering diagnostic information 564
 - initialization problems 564
 - log file example 565
 - overview 561
 - reservation objects 562
 - reservation styles 562

- RSVP agent problems *(continued)*
 - reservation types 561
 - service policy problems 565
- RSVP Agent, location of trace output 10
- RSVP processing 563

S

- SBCS (single-byte character set)
 - FTP client 336
 - FTP server 312
- SDUMP for FFST 15
- secure sockets layer (SSL) encryption 372
- sendmail, diagnostic aids for z/OS UNIX 393, 398
- server data block, displaying the TSDB 253
- server data extension, displaying the TSDX 254
- service level indication processing (SLIP) 11, 17
- service policy scopes
 - datatraffic 531
 - RSVP 532
- session problems
 - FTP 314
 - SNALINK LU0 hangs 404
 - SNALINK LU0 outages 405
 - tn3270 Telnet client hangs 374
 - tn3270 Telnet server hangs 368
 - tn3270 Telnet server outages 370
 - X.25 NPSI hangs 650
- SETPRINT, IPCS subcommand 272
- shortcut keys 747
- signals, name server 448
- simple mail transfer protocol, see also SMTP 383
- simple network management protocol, see also
 - SNMP 487
- simple network time protocol, see also SNTP 699
- single-byte character set (SBCS)
 - FTP client 336
 - FTP server 312
- SKMSG, IPCS subcommand 273
- SLA subagent
 - start options 490
 - statements 490
- SLIP (service level indication processing) 11, 17
- SMTP (simple mail transfer protocol)
 - abends 384
 - ADDRBLOK data set 388
 - defining 384
 - delivery problems 385
 - environment 383
 - incorrect output 386
 - looping problems 386
 - re-resolution of queued mail 387
 - receiver overview 383
 - RESOLVER trace 390
 - sender overview 383
 - spooling problems 384
- SMTP, location of trace output 9
- SNALINK LU0
 - abends 404
 - location of trace output 9
 - session hangs 404

- SNALINK LU0 *(continued)*
 - session outages 405
- SNALINK LU6.2
 - address space problems 420, 430
 - configuration mistakes 419
 - data corruption problems 428
 - data loss problems 427
 - DLC connection problems 422
 - location of trace output 9
 - network components 417
 - network problems 424, 426
 - traces 433, 438, 439
 - using NETSTAT 429
- SNAP (subnetwork access protocol) 718
- SNMP (simple network management protocol)
 - abends 491
 - agent 493
 - agent does not respond 503
 - client overview 488
 - communication with NCPROUTE 620
 - connection problems 495
 - I/O error using PING 505
 - incorrect output 498
 - interaction with NCPROUTE 619
 - MIB 487
 - multiple stack problems 494
 - NetView 488
 - osnmp command 488, 489, 507
 - osnmpd command 488
 - query engine 492, 493, 525
 - socket calls 490
 - starting with osnmp 507
 - subagent 490
 - traces 507, 508, 509
 - unknown variable in output 498
 - variable format incorrect 501
 - variable value incorrect 502
- SNMP agent
 - configuration data sets 489
 - security methods 489
- SNMP problems
 - related to NCPROUTE 622
 - SNMPIUCV subtask 492
- SNMP traces
 - agent trace example 510
 - query engine IUCV trace example 525
 - query engine trace example 512
 - subagent trace example 512
- SNMP, location of trace output 10
- SNMPIUCV subtask 492
- SNTP (simple network time protocol) 699
 - abends 699
 - debug trace 699
- socket calls, SNMP (simple network management protocol) 490
- socket control block, displaying 225
- socket data, tracing 135
- SOCKET, TCIPCS subcommand 225
- SPUFI 469
- SQL
 - problems with FTP 323, 346

- SQL (*continued*)
 - problems with NDB 467
- SSA REXX/370 runtime libraries 171
- SSL (secure sockets layer) encryption 372
- stand-alone dumps 13
- state, displaying TCP/IP 227
- STATE, TCPIPICS subcommand 227
- static routes compared to active routes 581, 719
- status, displaying device 32
- STDOUT, use by OMPROUTE 601
- storage map, displaying 213
- storage, displaying TCP/IP 227, 238
- STORAGE, TCPIPICS subcommand 238
- stream control blocks, displaying 239
- stream message block, displaying 273
- STREAM, TCPIPICS subcommand 239
- subagent programs, user-written for SNMP (simple network management protocol) 487
- subagent, OMPROUTE 490
- subagent, SLA 490
- subagent, SNMP (simple network management protocol) 490
- subnetwork access protocol (SNAP) 718
- subtasks, displaying TCP/IP 227
- SVC dumps 13
- SYSABEND dump 12
- SYSERROR data set 4
- syslogd
 - for diagnosing DNS problems 443
 - SNMP (simple network management protocol) agent traces 494
 - traces for FTP server 327
 - traces for OMPROUTE 601
 - traces for OROUTED 587
 - traces for REXECD 465
 - traces for RSHD 466
 - using with z/OS UNIX Telnet 349
- SYSMDUMP 12
- sysplex
 - addresses not returned 452
 - connection problems 453
- Sysplex Distributor 41
- SYSPRINT data set 4
- SYSTCPIS 136, 576
- system abends 21
- system dump header, displaying 209
- system trace 14
- SYSUDUMP 12

T

- task interface element (TIE) 676
- task related user exit (TRUE), CICS sockets 676
- tasks
 - stopping SNTPD
 - steps for 699
- TBEs (trace buffer entries), displaying 246
- TCA (trace control area), displaying 246
- TCB control blocks, displaying 241
- TCB, TCPIPICS subcommand 241
- TCBSUM, subcommand for TCIPICS 241

- TCP header fields, displaying 274
- TCP/IP
 - online information xix
 - protocol specifications 733
- TCP/IP socket, location of trace output 8
- TCP/IP subagent, statements 490
- TCPHDR, IPCS subcommand 274
- TCPIP state, displaying summary 227
- TCPIP storage, displaying summary 238
- TCPIP.DATA data set 310
- TCPIPICS subcommand
 - API 173
 - CONFIG 175
 - CONNECTION 177
 - COUNTERS 178
 - DUAF 192
 - DUCB 195
 - FIREWALL 200
 - FRCA 203
 - HASH 205
 - HEADER 209
 - HELP 210
 - LOCK 211
 - MAP 213
 - MTABLE 214
 - POLICY 215
 - PROFILE 217
 - PROTOCOL 219
 - RAW 222
 - REXX executables for 171
 - ROUTE 224
 - SOCKET 225
 - STATE 227
 - STORAGE 238
 - STREAM 239
 - symbols 173
 - syntax 171
 - TCB 241
 - TELNET 243
 - TIMER 244
 - TRACE 246
 - TREE 250
 - TSDB 253
 - TSDX 254
 - UDP 256
 - VMCF 258
 - XCF 260
- Telnet control blocks, displaying 243
- Telnet, displaying control blocks 243
- TELNET, TCPIPICS subcommand 243
- Telnet, tn3270
 - associating CTrace entry with client 373
 - client abends 374
 - client definitions 373
 - client overview 373
 - client session hangs 374
 - client trace example 377
 - commands and options 380
 - defining server 365
 - incorrect output from client 375
 - incorrect output from server 369

- Telnet, tn3270 *(continued)*
 - server abends 366
 - server logon problems 366
 - server overview 365
 - server session hangs 368
 - session outages 370
 - SSL encryption 372
 - starting client traces 376
- Telnet, z/OS UNIX
 - t -D all example 350
 - arrow keys do not respond 349
 - debug trace options 349
 - diagnostic messages sent to wrong file 349
 - improper authority error 349
 - locked keyboard 349
 - netdata and ptydata 350
 - utmp entries 364
- termination notification facility (TNF) 671
- TIE (task interface element) 676
- time and date formatting 276
- timeout
 - oping 29
 - PING 29
- timer control blocks, displaying 244
- TIMER, TCPIP subcommand 244
- TNF (termination notification facility) 671
- TOD, IPCS subcommand 276
- token-ring
 - defining for NCPROUTE 621
 - IEEE 802.5 716
- trace buffer entries (TBEs), displaying 246
- trace control area (TCA), displaying 246
- TRACE CT command 54, 70, 618
- trace output, location of
 - CICS 11
 - DDNS 9
 - Express Logon 11
 - FTP server 8
 - IMS 11
 - LPD server 8
 - LPR client 8
 - NCPROUTE 10
 - NDB 9
 - OMROUTE 10
 - OROUTED 10
 - OSF/Motif 10
 - Policy Agent 10
 - Popper 9
 - Resolver 11
 - RSVP Agent 10
 - SMTP 9
 - SNALINK LU0 9
 - SNMP 10
 - TCP/IP socket 8
 - TRMD 10
 - X.25 NPSI 11
 - Xwin 10
- TRACE RESOLVER, resolver 684
- TRACE, TCPIP subcommand 246
- traceroute function 40
- TRACERTE command 40

- traces
 - component trace 13
 - data trace 135
 - data traces 13
 - event trace for TCP/IP stacks 62
 - for FTP client problems 347
 - for FTP server problems 326
 - for SNALINK LU6.2 networks 433
 - GTF traces 14
 - master trace 14
 - OROUTED 588, 589, 590
 - packet trace 14, 406, 428, 437
 - SNALINK LU0 debug 413
 - socket API traces 74
 - system trace 14
 - TCP/IP internal 438
 - tn3270 Telnet client 376
 - tn3270 Telnet server 373
 - VTAM buffer 439
 - VTAM trace 14
 - z/OS UNIX application debug traces 14
- trademark information 752
- TRAPFWD trace example 530
- traps 11
- TREE, TCPIP subcommand 250
- TRMD problems, gathering diagnostic information 578
- TRMD, location of trace output 10
- TRUE (task related user exist), CICS sockets 676
- TSDB, TCPIP subcommand 253
- TSDX, TCPIP subcommand 254
- TSO console log from REXEC 455
- TSO DIG command 445

U

- UCBs (UDP control blocks), displaying 256
- UDP control blocks (UCBs), displaying 256
- UDP header, displaying 277
- UDP, TCPIP subcommand 256
- UDPHDR, IPCS subcommand 277
- user abends 21
- user interface
 - ISPF 747
 - TSO/E 747
- utmpx file 364

V

- VARY TCPIP command 28, 37
- VERBEXIT 17
- VIPA (virtual IP address) 585, 602
- virtual IP address (VIPA) 585, 602
- virtual machine communication facility (VMCF) 671
- VMCF (virtual machine communication facility) 671
- VMCF, TCPIP subcommand 258
- VTAM buffer trace
 - for tn3270 Telnet client session hangs 374
 - for tn3270 Telnet server session hangs 368
- VTAM trace 14
- VTAM, online information xix

W

WLM (workload manager) 452, 453
workload manager (WLM) 452, 453

X

X Window System

trace when XWTRACE=2 483

trace when XWTRACELC=2 484

X.25 NPSI

configuring for NPSI 645

configuring for VTAM 645

location of trace output 11

logon problems 649

overview 643

session hangs 650

XCF, TCPIP subcommand 260

Xwin, location of trace output 10

Z

z/OS, documentation library listing xxi

z/OS, listing of documentation available 743

Communicating Your Comments to IBM

If you especially like or dislike anything about this document, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this document.
- If you prefer to send comments by FAX, use this number: 1-800-254-0206
- If you prefer to send comments electronically, use this network ID: usib2hpd@vnet.ibm.com

Make sure to include the following in your note:

- Title and publication number of this document
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

z/OS Communications Server
IP Diagnosis Guide
Version 1 Release 4

Publication No. GC31-8782-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape



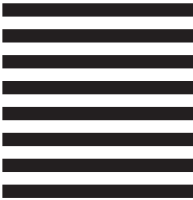
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Software Reengineering
Department G71A/ Bldg 503
Research Triangle Park, NC
27709-9990



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Program Number: 5694-A01 and 5655-G52

Printed in U.S.A.

GC31-8782-02



Spine information:



z/OS Communications Server

z/OS V1R4.0 CS: IP Diagnosis

Version 1
Release 4